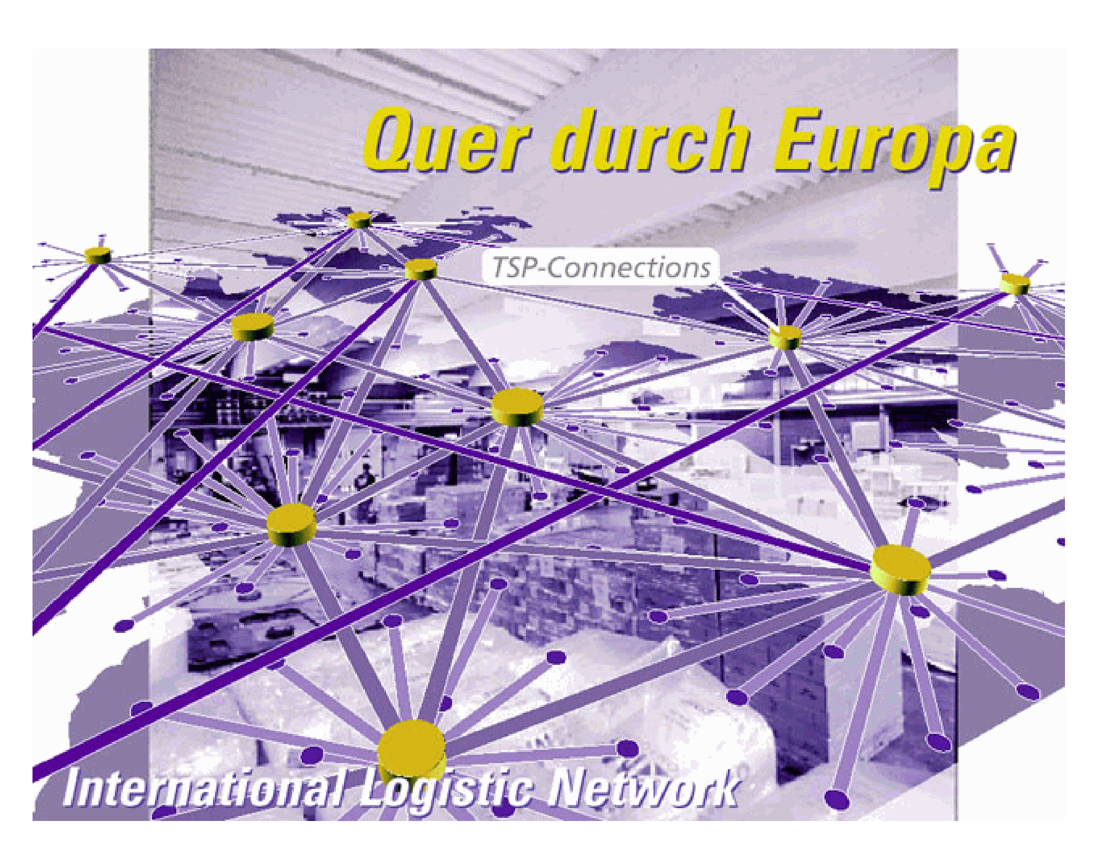
SNIFFER: Troubleshooting with the Sniffer

Sniffer 学习手册



编者：毛晋晋　著

深圳龙岗







绪　　言

最近沉下心来在学习 sniffer，对于 sniffer 没有什么好说的，大凡稍微懂点网络知识的或喜欢黑客的朋友都知道 sniffer=嗅探器,在这个方面相信很多人比我学的好，因为这个东西我接触的晚，而且网络这东西这也是最近两年才刚开始，所以很多东西还需要改进，前些日子在网上看了一篇《范伟导老师 Sniffer 课程资料》的贴子觉得很不错，感觉很深，实话直说这篇贴子我看了三遍，先介绍一下让我感兴趣的范老师（虽然我至今还没见过他照片当然本人更是没见过），而且在这篇教程中也借用许多范老师讲课的内容。

范老师，范伟导，现在是Sniffer中国技术中心的技术总监，中国唯一的Sniffer大师(SCM)，CISCO和Sniffer的授权讲师，这是范老师的邮箱：fanweidao@hotmail.com，大家没有什么特别的事不要轻易去打扰他。

我本人，毛晋晋，现在在深圳一家制造业公司做网络管理，2006 年获信息产业部的网络工程师中级职称，毕业后留校任教半年，现正在向CISCO方向学习，个人技术博客:http://blog.csdn.net/maodou521,MSN:maojinjincn@gmail.com 希望有这方面爱好的人一起来学习。

决定写 sniffer 教程有三个原因，一是因为个人最近在学习比较的清闲，第二方面就是前面说到的范老师的讲课给我很大的刺激,下面我到时会全文引用范老师一个学生写的范老师讲课内容。第三点是因为我自已也是一个搞网络的，毕竟这年头什么都要学点才不至于落伍嘛。

下面我稍微叙述一下 sniffer 教程的一些基本内容，以便让大家做个决定，哪些该看，哪些可以跳过去的，当然我希望大家能跳过去的越多越好，这样一来就表明大家的知识比起我来就丰富的多了,同时我这上面也有很多是从网上整理来的,所以大家的看时不要觉得奇怪,这里我在哪看过了,这里是什么地方来的,我现在申明就是这个意思。

首先，我要介绍给大家的就是范老师的那篇讲演稿，为什么会把这个做为开始呢，这是因为这篇稿恰恰是引领大家入门的一个好方法，在稿里范老师由浅入深的介绍了 sniffer 及为什么学网络的人要学，当然不单单是学网络的人要学，很多 IT 人都要学。

第二篇，主要讲讲我自已的一些感受，当然你也可以跳过。

第三篇，介绍嗅探的基本原理,从大体上了解一下它的运行环境,复习一下我们的基本网络知识,从大体上介绍 TCP/IP 体系,Sniffer 技术与 antisniffer 技术,这些都能很好的让我们了解 Sniffer 的发展,让大家从根本上明白这个东西.

第四篇，从大概上我介绍一下 sniffer 的产品及常见应用，这是一个开始，只有让大家熟悉了，大家才会从根本上去了解它。同时也让大家明白 sniffer 的一些功能和作用,当然同时还是复习了原理的东西.

第五篇，介绍 sniffer 的界面，大家在这一章节里可以自已动手实验一下，当然你要有条件才行。同时对一些基本的使用及案例进行分析,让大家有机会一边操作一边学习,更好的学习和在工作中使用 sniffer,体验 Sniffer 带来的乐趣,其中牵涉到的案例大家一定要动手去做,不然就没有效果了.

第六篇，谈及到了 TCP/IP，建议大家有可能将 TCP/IP 详解看看,当然可以让大家学习如何使用 Sniffer 学习 TCP/IP,同时让大家明白网络的运作原理等,希望能让大家受益.

第七篇，是关于嗅探的基本原理，也不能让大家快结束了，还不知道这个东西是怎么实现的。再次复习一次,在这里大约说了三遍,让大家加深印象,不然马上就忘了,那就不好了.

第八篇，Sniffer 的基本使用与实例，这一章节里就是真正的让你明白怎么去使用 Sniffer 了,再次实战操作了,希望大家好好利用最后一次操作的机会,记住:一定要动手去做!

最后介绍给大家的就是你用 Sniffer 之后，对得到的东西怎么去理解，这便就是数据分析实例,基本上就是按照一个循序渐进的过程,让大家从开始入门，再到最后，就是一个问为什么的过程，只要明白了这点，不管以后在什么样的工作中你都会相当不错。

最后感谢我的同事王志明先生,在整本书写作期间一直默默的提供实验环境,积极的加入到 Sniffer Pro 的各个测试过程及提供建议,在此表示感谢!

编者：毛晋晋
于深圳龙岗

目　录

范伟导老师 Sniffer 课程资料

言:

范老师现在是 Sniffer 中国技术服务中心的技术总监，是中国唯一的 Sniffer 大师（SCM），他有丰富的经验和经典案例，讲课讲得不错。

我是范老师的学生，我 2005 年学习了 Sniffer，发现收获很大，但我不能透露我的单位，因为我想范老师不会允许我把他的讲课内容公开。

以下内容是我根据上课录音编写的，基本上是范老师的原话。我整理了一个星期才整理出来，因为范老师在上课时有很多笔书，整理起来很困难， 有人会问，为什么不把录音共享出来，主要是课程中很多实验，只有录音，作用不大，我把他整理成文字，看起来会方便一些，当我全部整理完，估计可以出书了，版权费给谁呢？哈哈，我希望大家喜欢，如果反应良好，我把后面的内容也贴出来，很辛苦的，大家要珍惜。

大家不要放映，精华内容都在注释里。

大家有什么问题，可以发E-mail给范老师，fanweidao@hotmail.com，记住如果他问你是谁，你说是北京移动或广东移动或工商银行随便一个省分行的，因为这些单位的学生特别多，他肯定搞不清楚。哈哈，对不起了！范老师，我只是想帮你推广Sniffer.

课程内容：

大家好!欢迎大家参加 Sniffer 的认证课程!

先自我介绍一下！我叫范伟导，这是我的邮件(fanweidao@hotmail.com)，我现在没有工作（同学们：自由职业者）可以这么说。

介绍一下我的经历：毕业后我在一个台资电脑厂工作了一年，做硬件的。后来到了日本三洋工作，作 X400 的软件开发，做 ERP，用 RPG 开发，做了 4 年后来到了神州数码，作 CISCO 网络，原来在技术中心做实施，后来在培训中心做讲师，一共做了 5 年。

现在我是 CISCO 和 Sniffer 的授权讲师，不过现在我不想做 CISCO 了，想做 Sniffer，因为我觉得网络分析是一个很好的技术方向。等一下我还会跟大家聊一聊我们该往哪一个方向发展。

先看一下我们这个课程，这个课程事实上是两门课，第一门我们介绍怎样用 Sniffer 来做网络故障诊断，还有网络管理的一些方法和思路，第二门课我们介绍如何做应用的分析，这是 Sniffer 的新课程，我个人觉得非常好，以前的几个班学员也很喜欢。第一门课我们会讲 3 天，第二门课我们会讲 2 天。

接下来的半个小时我们不讲书本知识，讲讲我的经历和 Sniffer 究竟能用来做什么，我们为什么要学 Sniffer，其实我的目的是提起大家的学习兴趣，大家愿意学，我才讲的起劲。要不我一边讲，大家在噼噼啪啪上网，那我就讲不下去了（同学们笑）。

大家做网络都很多年了，想想我们以前的 10 兆以太网，现在的万兆以太网，想想 14.4k 的 modem,现在的 2M 宽带，以前的 x25,桢中继，现在的 SDH,MSTP,裸光纤。大家都经历这些，但我们才工作几年？就这几年，变化这么大，我不知道大家的工资有没有变化这么大,(同学们大笑），从 10 兆到万兆，1000 倍，几年工资张 1000 倍。有点难（同学们：不是有点难，是很难，不可能）。

再看看我们工作的变化，以前能配配路由器就很牛了，现在似乎谁都会了，记得几年前，我帮一个小集成商配一台 4000 系列交换机，收了 2000 元，15 分钟搞定。（同学们：好爽，介绍一些给我们做），没有了说说你们的工作。平常工作中做些什么（同学们：做做网线，杀病毒，帮领导装机器）大家想想，这是我们想做的工作吗，以前这些都不用我们做，现在大家感觉是不是地位在下降，工资也不涨，好歹我们也是蜘蛛级的人物呀，不是有个笑话说蜜蜂是空姐，做网络的是蜘蛛吗。（同学们笑）

我们该怎么办?

现在说说我的观点，我们都希望工资能年年涨，不要求 1000 倍，（同学们：不要求那么高，一年 20%就行了），20%？不止吧，从毕业到现在，你们工资不止年均涨不止 20%吧。 （同学们：我们不能跟您比）也有可能，你们的起点高，我毕业的时候才有 650 元。

大家回顾一下，做 IT 的谁的收入高？
1、销售
2、领导
3、咨询专家
4、售前工程师
5、售后工程师

我们在座的有 3 个是网络中心的主任或科长，他们的收入肯定比一般工程师高，我祝愿你们步步高升，收入节节高。在座大多数是网络工程师，我们该怎么走，其实你们现在的单位都很好，但将来怎样很难知道，比如前几年银行的收入令人羡慕，现在他们却担心降工资，现在移动的收入不错吧，我有汽车厂商的学员，他告诉我他们的收入比移动好点，（同学们：哇）这是他们自己说的，好多少就没说了，还有某政府单位的，什么单位不便说，他们没告诉我他们的收入，只说，价格少于 4 万的笔记本他们不用，哇靠，4 万的笔记本，什么配置？（同学们：那是服务器）我们不能比，人比人，气死人。我们没法进入这些公司的，还是脚踏实地一点好，但我们做技术的也要考虑如何提高我们的收入，做技术的要提高收入，地位是关键，前面大家说只做做线，杀杀病毒，我们的地位在下降，工资怎么长得起来呢？想想我们做技术的，谁的收入高，做数据库的比做服务器的高，为什么 Oracle 那么火，做服务器的比写程序的高，写程序的比做网络高，

这是普遍现象，不说特殊情况。其实大家发现一个特点没有，凡是掌握企业关键业务的收入都很高，你看作数据库的，数据库坏了，企业完蛋了，领导当然重视，现在不仅讲存储，还讲灾备，你看很多银行，北京一个数据中心，上海一个数据中心。

我们网络怎样，设计的都是高可靠性的端到端备份，出问题的机会很少，而当应用出什么问题，都说是网络问题。举个例子，有个单位（税务的学员告诉我的），有一天应用突然变慢，大家都说网络慢了，我们用尽 troubleshooting 的技术也发现不了问题，结果作数据库的工程师偷偷改一下表空间，好了，没问题了，我们不知道怎么好了，做数据库的不说他们有问题，还说网络好了，领导问我网络怎么好的，我不知道呀，领导说：赶快查出原因，避免再出现类似问题，哇塞，怎么查，本来网络就没问题，查什么查。（同学们笑）

所以现在大家用一个字来形容我们的工作？你们会用什么字（同学们：累、苦）很贴切，苦、累所以我们不能一直停留在网络的 troubleshooting,我们必须提高我们的地位，要不我们会累死。

怎么提高地位，我们必须了解我们的业务，也就是要了解应用，了解应用在我们网络上的行为特征，很重要的一个词行为特征。当我们了解了业务的行为特征，我们能定位某一个问题的真正故障点，举个例子：网络应用变慢，可能的原因有什么？网络问题，服务器问题，数据库问题，应用程序问题，客户机问题。如果我们能够判断是哪一部分问题，我们就有发言权了，比如说刚才那种情况，如果我们直接说这是数据库问题，不是网络问题，领导会问，你凭什么说是数据库问题，你可以拿出 Sniffer，专家系统上写着，DB Slow Server response 诊断，（范老师在演示）再看解码，做一个用户验证操作，花了 1.731 秒，有根有据，大家想一想，有了 Sniffer 我们可以了解我们的业务行为特征，可以排除我们的责任，不但工作轻松了，地位也提高了。（同学们笑）

以前我们应用出现问题的时候我们总是分头查找问题，结果往往是没有结果，因为这种查找方式范围太大了，我们做 troubleshooting 第一步应该是：隔离故障。

如果我们有了 Sniffer，首先用 Sniffer 看一下，最有可能是哪一部分问题，再安排检查，这样不但节省人力，速度会更快，效率也更高。

如果有人问我们Sniffer是什么？大家都会说是协议分析仪，你看sniffer网站（www.networkgeneral.com）上说的是应用和网络分析系统。究竟Sniffer是什么样的一个东西，我们要了解他的发展过程。其实很多类似的产品比如ethereal,netscout,wildpacket等都有类似的发展过程。

第一阶段是抓包和解码，也就是把网络上的数据包抓下来，然后进行解码，那时候谁能解开的协议多，谁就是老大，Sniffer 当时能解开的协议最多，也就理所当然地成了老大，现在 Sniffer 能解开 550 种协议，还是业界最多的，

第二阶段是专家系统，也就是通过抓下来的数据包，根据他的特征和前后时间戳的关系，判断网络的数据流有没有问题，是哪一层的问题，有多严重，专家系统都会给出建议和解决方案，现在 Sniffer 的专家系统还是业界最强的

第三阶段：是把网络分析工具发展成网络管理工具，为什么要这样，如果 Sniffer 知识用作网络分析，那 Sniffer 的软件就够用了，现在软件的 portable 基本上都是盗版的，sniffer 没钱赚了，所以它必须往网络管理方向转，要作为网络管理工具，就必须能部署在网络中心，能长期监控，能主动管理网络，能排除潜在问题，要做到这些，就要求有更高的性能，所以 Sniffer 就有了相应的硬件产品，比如说分布式硬件平台，InfiniStream 等，我知道在座各位都买了 Sniffer 的硬件，这时候如果用软件的 Sniffer 性能就不行了。

我们看一下，Sniffer 究竟有什么用？

第一，Sniffer 可以帮助我们评估业务运行状态，如果你能告诉老板说，我们的业务运行正常，性能良好，比起你跟老板报告说网络没有问题，我想老板会更愿意听前面的报告，但我们要做这样的报告，光说是不行的，必须有根据，我们能提供什么样的根据呢。比如各个应用的响应时间，一个操作需要的时间，应用带宽的消耗，应用的行为特征，应用性能的瓶颈等等，到第二门课，我会告诉大家怎么做到有根有据。

第二，Sniffer 能够帮助我们评估网络的性能，比如，各连路的使用率，网络的性能的趋势，网络中哪一些应用消耗最多带宽，网络上哪一些用户消耗最多带宽，各分支机构流量状况，影响我们网络性能的主要因素，我们可否做一些相应的控制，等等。

第三，Sniffer 帮助我们快速定位故障，这个大家比较有经验，我们记住 Sniffer 的三大功能：monitor,expert,decode 这三大功能都可以帮助我们快速定位故障，我后面通过案例演示给大家看，大家再做做实验，很快就上手了（同学问：范老师，是否要学 Sniffer 必须对协议很熟，）不一定，我们可以通过 Sniffer 来学习各种协议，比如 ospf,以前学网络的时候，讲 OSPF 的 LSA 好像很复杂，你用 Sniffer 看看，其实他的协议结构还是不复杂的，一般情况下，我会要求学 Sniffer 的学员有 CCNP 的基础，或者有几年的网络管理经验，我自己也是这样，刚开始只是用 Sniffer 抓抓包，抓下来也不知道怎么分析，当我学完 CCNP 后，学了 CIT，以为自己不错了，会排除很多网络故障，但实际上很多问题我还是解决不了，比如网络慢，他又不断，断了我很快能解决，网络慢，或者丢包，一般的排错知识还是很难的，那时候开始学 Sniffer，才发现很好用。

第四，Sniffer 可以帮助我们排除潜在的威胁，我们网络中有各种各样的应用，有一些是关键应用，有一些是 OA，有一些是非业务应用，还有一些就是威胁，他不但对我们的业务没有帮助，还可能带来危害，比如病毒、木马、扫描等，Sniffer 可以快速地发现他们，并且发现攻击的来源，这就为我们做控制提供根据，比如我们要做 QOS，不是说随便根据应用去分配带宽就解决了，我们要知道哪一些应用要多少带宽，带宽如何分配，要有根有据。我们再回过头看一下 Sniffer 什么时候开始流行的，再 2003 年冲击波发作的时候，很多 Sniffer 的

用户通过 Sniffer 快速定位受感染的机器，后来很多人都知道 Sniffer 可以用来发现病毒，Sniffer 的知名度暴涨，盗版用户也暴涨（同学们大笑），后来震荡波发作的时候，很多人用 Sniffer 来协助解决问题。我想强调的是 Sniffer 不是防病毒工具，这也只是他的一个用途，而且只对蠕虫类型对网络影响大的病毒有效，对于文件型的病毒，他很难发现。

另外要说明的事，Sniffer 还可以用来排除来自内部的威胁，现在我们网络中有各种各样的网络安全产品，防火墙、IDS、防病毒软件，他们都有相应的功能，但真的有效吗，能解决全部威胁吗，我们要进行评估，用 Sniffer 就能评估内网的安全状况，有没有病毒，有没有攻击，有没有扫描，像防火墙、IDS、防病毒软件他们都是后知后觉的，它必须有特征才能阻绝，而 Sniffer 是即时监控的工具，通过发现网络中的行为特征，判断网络是否有异常流量，所以 Sniffer 可能比防病毒软件更快地发现病毒。我在神州数码的时候，冲击波震荡波都是我先发现的，有趣的是当时我都在上 Sniffer 的课，中午休息，我把 sniffer 驾到公司网络，再 Hosttable 看到广州一台机器很多广播，接着广州另外一台机器也开始发广播，接着深圳也感染了，我马上通知 IT 管理人员，他们把这几台机器断网，后来才知道有冲击波病毒，防病毒软件还不能杀。

刚才讲到异常流量，这是一个很重要的概念，什么是异常流量？我们怎么判断是否异常，这又涉及另外一个概念，叫基准线分析，什么是基准线，基准线是指我们网络正常情况下的行为特征，包括利用率、应用响应时间、协议分布，各用户贷款消耗等，不同工程师会有不同基准线，因为他关心的内容不同，只有知道我们网络正常情况下的行为特征，我们才能判断什么是异常流量。
第五，做流量的趋势分析，通过长期监控，可以发现网络流量的发展趋势，为我们将来网络改造提供建议和依据

第六点就是应用性能预测，这点很有用，会用的人不多，我们第二门课会讲，Sniffer 能够根据捕获的流量分析一个应用的行为特征，比如，你现在有一个新的应用，还没有上线，我能评估他上线后的性能，比如在用户在网络中心有多快，用户在省中心有多快，用户在市中心有多快，都可以提供量化的预测，准确率挺高的，误差不超过 10%。我们还可以用她来评估应用的瓶颈在哪，不同应用瓶颈不同，比如有些应用慢了，增加网络带宽效果很明显，比如 FTP 这种应用，有些应用慢了增加带宽没什么效果，比如 TELNET 应用，我们还可以预测网络带宽增加的效果，比如我将 2 兆提高到 8 兆应用性能有多大的提升，Sniffer 能比较准确地预测。

在这里我们提到三个重要概念，网络行为特征，异常流量，基准线，大家理解了吗？在这里，我不想太多介绍产品，我不是来推销 Sniffer 的(同学们笑)，我们主要探讨网络分析技术。

Sniffer 的便携式就是我们用的那种盗版软件（同学们笑），我不用介绍了，这门课我们用 Sniffer 的便携式来讲，因为分布式和 InfiniStream 也有一样的界面，上课的时候我们都是用便携式。

Sniffer 的分布式包括 4100 和 6040，主要是放在网络核心可以长期监控、分析，4100 可以处理千兆流量，6040 可以处理 8 千兆流量，这是业界性能最高的产品。

Sniffer 的 InfiniStream 的特点是可以长期抓包，最多有 4 个 T 的存储空间，可以长期抓包，可以进行回溯性分析，这点对有些用户来说很重要，比如今天早上 10 点半，某个应用很慢，十分钟后又正常了，如果没有 InfiniStream,流量没有保存，我们就很难分析问题在哪，如果有了 InfiniStream,这些流量都会保存下来，自动的，就是长期抓包，我们就可以找出当时的流量，进行分析，一个很好的设备，现在支持 1800 兆线速捕获，这也是其他厂商没有的。

这些你们买设备的时候代理都给你们说清楚了，我就不多讲了。

怎么样，听了这么久，感觉如何？有兴趣吗？

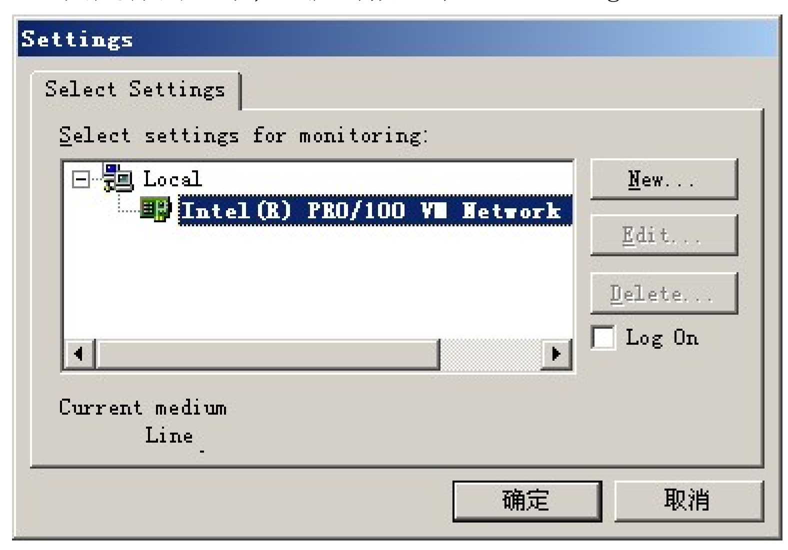
其实我个人是很喜欢 Sniffer 的，我当时从三洋出来的时候，错过了去 IBM 的机会，去了神州数码才知道，他们 IBM 需要做 X400 的人，去了神州数码，老板问我想做网络还是想做主机，我说做网络吧，那时一个 CCIE 二三十万的收入是有的，结果到我考过 CCIE 笔试的时候，CCIE 就值 10 万吧，真是绝望了，（同学们笑）为什么当时不做主机呢，我那些做 6000 的同事现在都不错，又不累。做网络不就一个字：累吗（同学们笑）我也没有去考实验了，01 年的时候我考了 CCSI，就是 CISCO 授权讲师，后来讲了很多 cisco 的课，我 CISCO 的学员有 1000 个，后来又讲 Sniffer 的课，Sniffer 的学员有 300 个，我的学员不少，有不少关系不错的，他们过的比我好（同学们笑，你做讲师也不错呀，收入不低吧），以前讲 CISCO 的时候收入不高,一天也就 1000 到 1500,讲 Sniffer 会好一些,(同学们：讲 Sniffer 一天多少）这还不好公开，如果你们有兴趣开班，我们再聊（同学们笑）。后来我考过了 SCM，Sniffer 的最高级认证，叫 Sniffer 大师。中国就我一个人，（同学们：哇，难不难考，考几门），Sniffer 的考试不难，这个我后面会讲，考过 SCM 的全球也只有 62 个，亚太区只有 5 个，所以 Sniffer 原厂的课程都是我讲的。在今年，我会去协助组建 Sniffer 中国技术服务中心，以后你们有什么问题都可以联系我，我在那里是技术总监。我们还有在各行各业的 Sniffer 专家小组，都是喜欢使用 Sniffer 的人在一起交流使用心得，分享一些案例，你们如有兴趣，到时我可以邀请你们参加，不过首先要认真听课。（同学们笑）

好了，讲了这么多，目标只有一个，提起大家的学习兴趣，接下来讲课程的内容，首先把 Sniffer 打开。

好，大家都打开了吗，有问题随时告诉我，（跑去解答问题去了）如果大家在上课的时候有任何疑问，随时可以打断我，不用给我面子，我也不一定能回答所有问题，不过没关系，交流总会进步的。

好，我们继续第一个我要介绍的是 local agent。什么叫 local agent，大家打开菜单“File->select settings”，这时候，大家可能只看到一个 local 下

面是你的网卡,这就叫做一个“local agent”。



事实上，一个“local agent” 就像一个探针。我们知道 Sniffer 的工作原理很简单，就是把网卡设成混杂模式（叫做 promiscuous），所谓混杂模式，就是把所有数据包接受下来放入内存，大家知道一般情况下,PC 机只接受目的 mac 地址为自己网卡或广播、组播的数据包。sniffer 就是这样把所有数据包都接收下来，在进行分析。

大家看我这里有多个 agent,怎样可以做多个 agent 呢，可以不同网卡做不同的 agent，就像你们的分布式 sniffer 一样，有多个网卡，那就是多个 agent,infinistream 也一样。

其实一个网卡，也可以做多个 agent,大家试一下，new 一个，给他加上说明，就叫 101 把，选中你们的网卡，下面选 no pod，copy setting 留空，那个 pod 是你外接 sniffer book 时候用的。大家看看你们的 agent 多了一个，101 括号 local_2。对不对（同学们：对）

好，不错。

我们为什么建立多个 agent 呢。不同的 agent 可以定义不同的阀值，可以有不同的过滤器，可以有不同的触发器，不同的地址本。

比如说，你们有一台笔记本装着 sniffer，大家都用它，那不同的工程师可以自己定义一个 agent，自己定义自己的过滤器，互不干涉，比如不同的网段有不同的阀值，也可以定义不同的 agent。

那 agent 的参数保存在哪里呢，大家打开 c:\program files\nai\sniffernt\program,大家看到 local local_2,这就是两个不同的 agent 保存参数的地方。大家看到两个 CSF 文件，一个是 sniffer.csf，另外一个是 Snifferdisplay.csf。这是过滤器文件，当我们使用 sniffer 一段时间之后，大家会累积许多好的过滤器，一定记得保存下来，就是把这两个文件考出来就行了，如果你看到别人那里有好的过滤器，也可以拷过来。不过当你要倒回去

的时候，4.8 比较好办直接倒入就行了，4.75 比较麻烦，我后面讲定义过滤器的时候再教大家。过滤器是 sniffer 最难、最有意思、最重要的一部分，大家放心，我能让大家成为高手。（同学们笑）

好，“local agent”讲完了，local agent 是什么？事实上就是定义一个环境变量，不同的环境不同的参数。

好，休息一下，待会儿讲 monitor 功能。

中间休息的时候，我问了范老师一个问题：我看书上说，TCP 是可靠的,UDP 是不可靠的，那要不可靠 UDP 来干什么？（各位：我的问题是不是很傻？但我确实不知道呀）

范老师：不错，这个问题非常好！（嘿嘿！）

TCP 叫传输控制协议，他的特点是：有连接，有流控，有顺序号/确认号，开销比较大，一般是 20 个字节的头。

UDP 叫用户数据报协议，开销小，8 个字节的头，无可靠保证。
我后面有详细介绍 TCP 和 UDP，我们先看您的问题。

首先，UDP，不可靠，是指，在传输层不提供可靠保证，并不意味着所有使用 UDP 的应用都不可靠。

我们来比较几个应用（范老师用他的 trace file 给我演示）

DNS，53 端口，进行查询时，用的是 UDP，因为要求速度快，比如我要查 networkgeneral 的地址，你只要告诉我 ip 是多少就行了，如果要进行 3 次握手建立连接，再去取到 IP，那就慢了，所以用的是 UDP,一个字：快。没响应怎么办，事实你看（他在演示）它会同时向多个 DNS 查询，所以没响应也没关系，你看这个响应名字错误，找不到。所以 UDP 还是有用的，特别是像 DNS 查询这种应用，丢了也就丢了，我再查。但 DNS 也有用 TCP 的时候，比如 DNS 服务器的同步，用的就是 TCP 的 53 端口

TFTP，您所了解的 TFTP,用的是 UDP 吧，他不可靠吗，事实上文件传输,必须保证可靠。不但要保证能知道丢包重传，还要有顺序号，应付错序到达的情况，也就是我们常说的后发先至。事实上 TFTP 是怎样工作的，你看（他在演示），每一个数据块都有 Id 号，一块 512 字节，一次传输，一次确认，这就相当于 TCP 的顺序号和确认号。所以 UDP 是不可靠的,但很多使用 UDP 协议的应用是可靠的,只是在应用层去保证可靠性，很多人说用 UDP 效率高，事实上 TFTP 在传输大文件的时候,比 FTP 效率更地，我们后面有专门的实验。

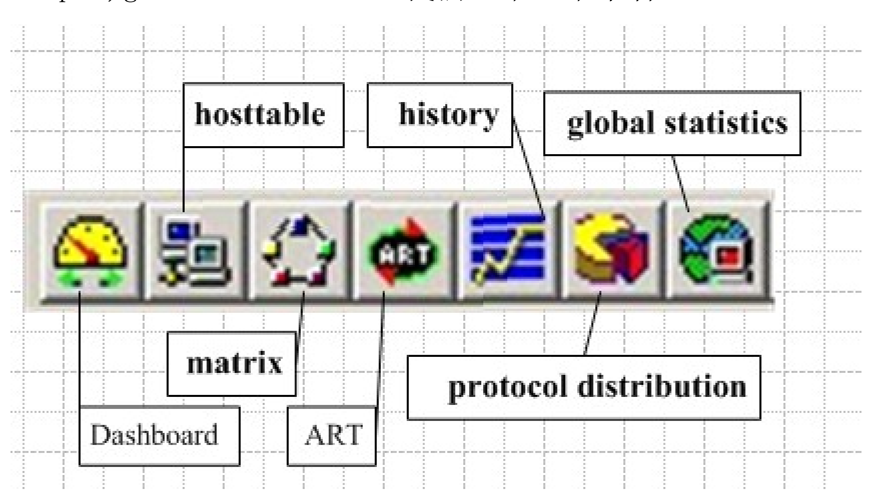
视频流量，（没有演示）对于视频流量，也是需要可靠保证的，但要求不是很高，所以不会像 TFTP 那样每一个数据报都确认，而是传多个数据包确认一

次，要不效率就太低了，究竟多少个数据包确认一次，开发人员需要不断测试。我的解释清楚吗，（我说：明白了！谢谢！）

（确实看着演示，很容易就理解了，中间我们有许多对话，我省略了，确实如果只听录音是不明白的，这是我为什么要整理成文字给大家看，好累呀！大家给我加油！）

好，我们继续！

我们来看一下 Sniffer 的七大 monitor 功能，有 Dashboard,hosttable,matrix,ART，protocol distribution,history sample,global statistics 我们一个一个来看，



先看 dashboard。



这个大家很熟悉了，我不用多讲，dashboard 有 3 个仪表，分别是使用率，每秒钟包数量，每秒钟错误率，下面都有两个数字，前面一个表示当前值，后面一个表示最大值。

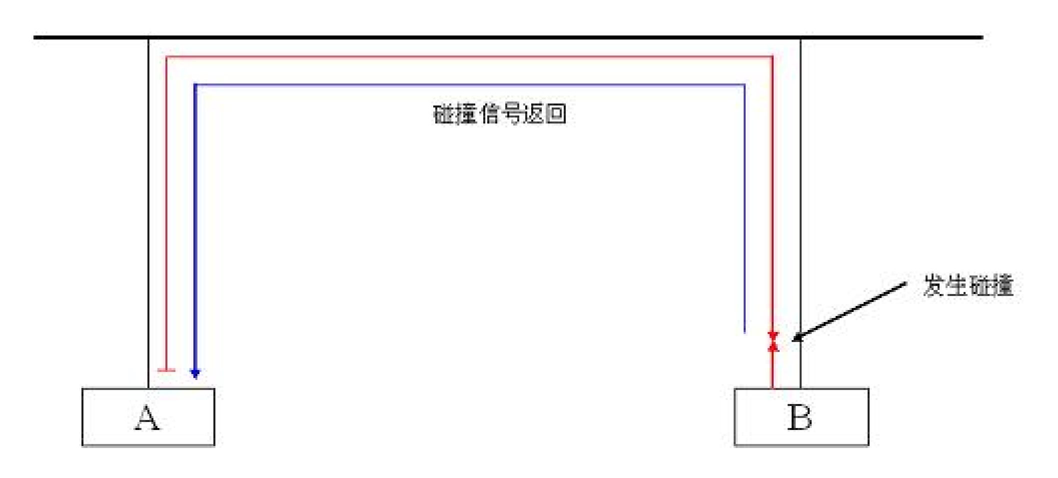
下面还有 long term,和 short term,Long term 每 30 分钟采样一次，一共可以采样 24 小时，short term 每 30 秒钟采样一次，可以采样 25 分钟大家自己试一下，首先把 File 里面的 loopback 选上，这样我们发的数据包就不会发到网络中去，然后打开 101 目录里的 TCPdemo7a 那个 trace file ,再用 packet general 发包，选 send current buffer，连续发送。（我们是跟着范老师做的）。

好了，大家试了一遍，感觉应该是一样的，就是这有什么用？没用，对吧，我也这样觉得（同学们笑）但如果你要监控某一台服务器的时候，这个是有用的，比如你把一台服务器的接口 monitor 过来，这样你就可以看到这台服务器的流量状况了，这就是一个很好的基准线呀。当然大家用的是硬件产品，就更方便了。大家注意到下面还有错误报的统计，要注意的是一般的网卡是抓不了错误包的，要用专用网卡，一块网卡上万块，NG 好黑呀（同学们笑）其实大家知道通过交换机的存储转发，基本上很少错误包，所以不用关注它。

在这里我想解释一下以太网的错误包，这对大家学习网络是很有帮助的，特别是了解一下封装的概念。

（请看下一页：以太网为什么要 64 个字节）

以太网帧为什么最小要64个字节



(这是范老师的板书,我画不出来,大家将就点吧)
(这是范老师的板书,我画不出来,大家将就点吧)

以太网是无连接的，不可靠的服务，采用尽力传输的机制。以太网 CSMA/CD 我就不多讲了，我相信大家都了解这个原理。

以太网是不可靠的，这意味着它并不知道对方有没有收到自己发出的数据包，但如果他发出的数据包发生错误，他会进行重传。以太网的错误主要是发生碰撞，碰撞是指两台机器同时监听到网络是空闲的，同时发送数据，就会发生碰撞，碰撞对于以太网来说是正常的。

我们来看一下，假设 A 检测到网络是空闲的，开始发数据包，尽力传输，当数据包还没有到达 B 时，B 也监测到网络是空闲的，开始发数据包，这时就会发生碰撞，B 首先发现发生碰撞，开始发送碰撞信号，所谓碰撞信号，就是连续的 01010101 或者 10101010,十六进制就是 55 或 AA。这个碰撞信号会返回到 A，如果碰撞信号到达 A 时，A 还没有发完这个数据包，A 就知道这个数据包发生了错误，就会重传这个数据包。但如果碰撞信号会返回到 A 时，数据包已经发完，则 A 不会重传这个数据包。

我们先看一下，以太网为什么要设计这样的重传机制。首先，以太网不想采用连接机制，因为会降低效率，但他又想有一定的重传机制，因为以太网的重传是微秒级，而传输层的重传，如 TCP 的重传达到毫秒级，应用层的重传更达到秒级，我们可以看到越底层的重传，速度越快，所以对于以太网错误，以太网必须有重传机制。

要保证以太网的重传，必须保证 A 收到碰撞信号的时候，数据包没有传完，要实现这一要求，A 和 B 之间的距离很关键，也就是说信号在 A 和 B 之间传输的来回时间必须控制在一定范围之内。IEEE 定义了这个标准，一个碰撞域内，最

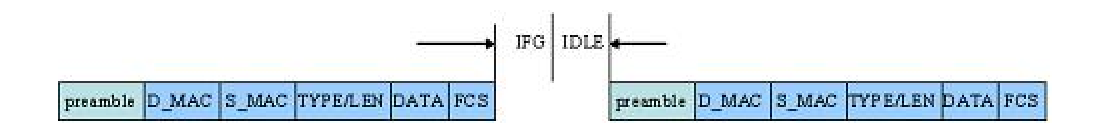
远的两台机器之间的 round-trip time 要小于 512bit time.(来回时间小于 512 位时，所谓位时就是传输一个比特需要的时间）。这也是我们常说的一个碰撞域的直径。

512 个位时，也就是 64 字节的传输时间，如果以太网数据包大于或等于 64 个字节，就能保证碰撞信号到达 A 的时候，数据包还没有传完。
这就是为什么以太网要最小 64 个字节，同样，在正常的情况下，碰撞信号应该出现在 64 个字节之内，这是正常的以太网碰撞，如果碰撞信号出现在 64 个字节之后，叫 late collision。这是不正常的。

我们以前学习 CISCO 网络的时候，CISCO 交换机有一种转发方式叫 fragment-free，叫无碎片转发，他就是检查 64 个字节之内有没有错误，有的话不转发，这样就排除了正常的以太网错误包。

(这是范老师的板书,我画不出来,大家将就点吧)
我们再来看一看以太网的帧结构。

以太网帧结构



要讲帧结构，就要说一说 OSI 七层参考模型。七层参考模型大家很熟悉，以前我们看书的时候会觉得不知所云，我刚学的时候就是这感觉，其实我们只要掌握两点就行了。

一个是访问服务点，每一层都对上层提供访问服务点（SAP），或者我们可以说，每一层的头里面都有一个字段来区分上层协议。

比如说传输层对应上层的访问服务点就是端口号，比如说 23 端口是 telnet，80 端口是 http。IP 层的 SAP 是什么？（同学们没说话）。

其实就是 protocol 字段，17 表示上层是 UDP，6 是 TCP,89 是 OSPF，88 是 EGIRP,1 是 ICMP 等等。

以太网对应上层的 SAP 是什么呢？就是这个 type 或 length。比如 0800 表示上层是 IP，0806 表示上层是 ARP。我后面还会将各种以太网的帧类型。

第二个要了解的就是对等层通讯，对等层通讯比较好理解，发送端某一层的封装，接收端要同一层才能解封装。

我们再来看看帧结构，以太网发送方式是一个帧一个帧发送的，帧与帧之间需要间隙。这个叫帧间隙 IFG—InterFrame Gap

IFG 长度是 96bit。当然还可能有 Idle 时间。

以太网的帧是从目的 MAC 地址到 FCS,事实上以太网帧的前面还有 preamble，我们把它叫做先导字段。作用是用来同步的，当接受端收到 preamble，就知道以太网帧就要来了。preamble 有 8 个字节前面 7 个字节是 10101010 也就是 16 进制的 AA，最后一个字节是 10101011,也就是 AB，当接受端接受到连续的两个高点平，就知道接着来的就是 D_mac。所以最后一个字节 AB 我们也叫他 SFD（帧开始标示符）。

所以在以太网传输过程中，即使没有 idle，也就是连续传输，也有 20 个字节的间隔。对于大量 64 字节数据来说，效率也就显得不高。

所以，有时我们用下载数据来检查我们的网速，这是不完全准确地，我们要了解他的传输特征，才能准确判断电信究竟给了你多少带宽。我有一个移动的学员，他说用户总怀疑我给他的带宽不够，其实我肯定给他两兆了，所以有时运营商也挺不容易（同学们笑）。后来我告诉他怎么样用 sniffer 来测带宽，不知道他后来成功了吗，我没有得到反馈。后面我会介绍怎样用 Sniffer 来做带宽测试，非常精确的喔。我给很多用户作过带宽测试，他们大多都是怀疑电信给的带宽不够。（同学们问：有没有不够的时候？）我测试的案例里还没有。还有就是帮集成商作方案验证，比如，集成商给用户作了多链路捆绑，或路由负载均衡，用户说比原来更慢了，我去证明给用户看，负载均衡确实做起来了，流量分担很正常。（同学们问：那为什么会慢呢），这就涉及到应用的特征和不同厂商采用均衡的机制。我还没试过作进一步分析。因为这是集成商的朋友叫我去帮忙的，我只要证明给用户看方案没问题，并告诉集成商如何给用户解释就行了，在做下去，就会画蛇添足了，因为可能让用户觉得我的水平比我朋友高，那不是帮倒忙了。（同学们笑）所以帮忙也要适可而止。 （同学们笑）

好了，有点扯远了。前面讲这些主要是帮大家复习以下以太网知识，大家别担心，时间是足够的，因为这门课里有很一些基础的知识，比如交换原理、vlan 原理，那些知识我都会跳过，我第一天的内容不会很难，考虑到大家远道而来，第一天都很累。但后面回越来越难，大家要有心理准备。晚上要早点睡觉（同学们笑）。还有一个，就是大家别指望能记得住我讲得全部内容，今天讲得明天还记得一点，后天就全忘了，（同学们笑），到了课程结束的时候，基本上全忘光了，（同学们大笑），所以做笔记很重要，我建议大家把笔记写在书上，到时才对得起来。我也注意到一些同学在录音，我知道的，不用放在桌子底下(同学们笑），那样效果不好，（同学们大笑），其实这是不允许的，不过没关系，只有一个要求，不要放在互联网上。

（编者：写到这里，有点写不下去了，觉得很内疚，觉得对不起范老师。我参加过很多培训，范老师是我很喜欢的一个老师，他讲课不会非常幽默，但很实用，这是因为他有很多经历，他在讲课过程中，会补充很多课程以外的东西，比如很多网络中的细节知识，很多工作中的思路，我觉得这方面收获很大，我个人觉得是对我知识的全面补充，学完之后觉得不仅学会了 Sniffer，网络管理的思路更清晰了，现在我指导工程师时，套了很多范老师的话，我觉得范老师很好。怎么办？我在进行思想斗争。。。该不该再写下去。我想在论坛里发起投票，听听大家的意见，我该不该再写下去。）

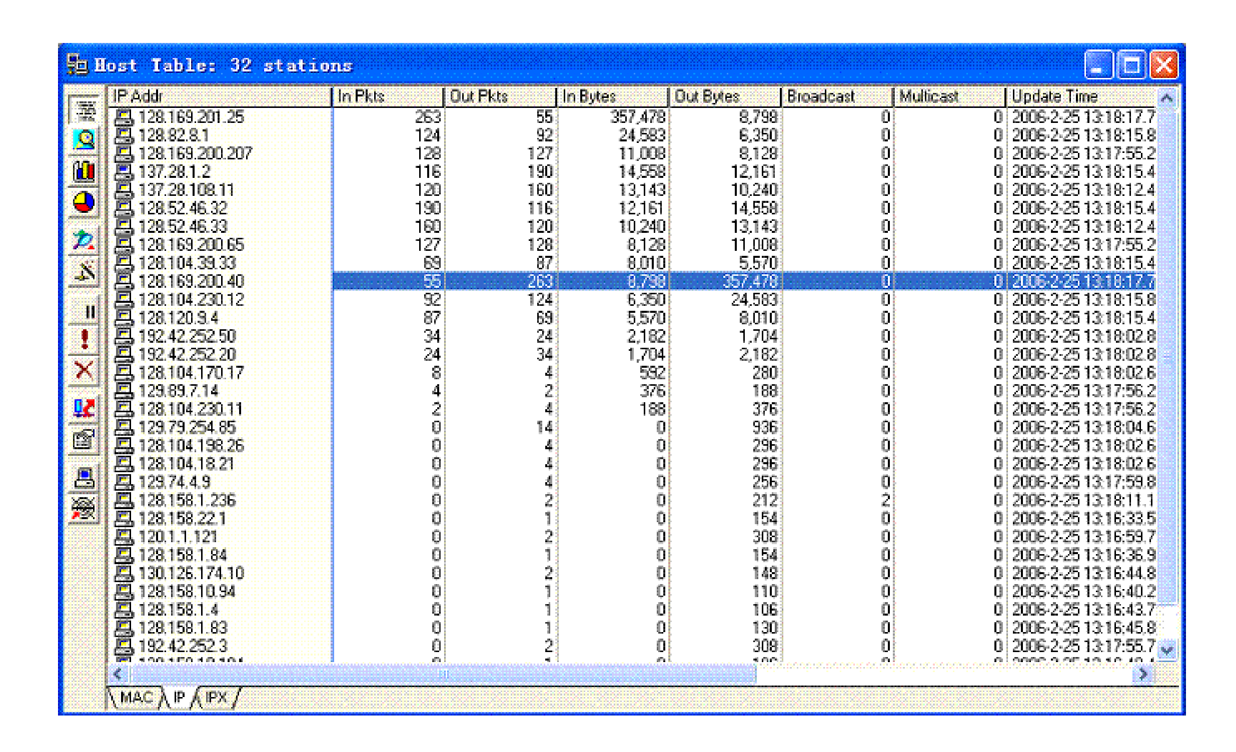
（编者：范老师的课程内容： 第一天 monitor 功能，Sniffer 的部署 。

第二天 expert，capture filter ，troubleshooting

第三天 decode，display filter ，trigger

第四天 应用的类型，应用的剖析，应用的分析思路

第五天 应用性能的分析，应用性能预测）


Host Table: 32 stations

IP Addr	In Pkts	Out Pkts	In Bytes	Out Bytes	Broadcast	Mulicast	Update Time
128.169.201.25	263	55	357,478	8,798	0	0	2006-2-25 13:18:17.7
128.82.8.1	124	92	24,583	6,350	0	0	2006-2-25 13:18:15.8
128.169.200.207	128	127	11,008	8,128	0	0	2006-2-25 13:17:55.2
137.28.1.2	116	190	14,558	12,161	0	0	2006-2-25 13:18:15.4
137.28.108.11	120	160	13,143	10,240	0	0	2006-2-25 13:18:12.4
128.52.46.32	190	116	12,161	14,558	0	0	2006-2-25 13:18:15.4
128.52.46.33	160	120	10,240	13,143	0	0	2006-2-25 13:18:12.4
128.169.200.65	127	128	8,128	11,008	0	0	2006-2-25 13:17:55.2
128.104.39.33	69	87	8,010	5,570	0	0	2006-2-25 13:18:15.4
128.169.200.40	55	263	8,798	357,478	0	0	2006-2-25 13:18:17.7
128.104.230.12	92	124	6,350	24,583	0	0	2006-2-25 13:18:15.8
128.120.9.4	87	69	5,570	8,010	0	0	2006-2-25 13:18:15.4
192.42.252.50	34	24	2,182	1,704	0	0	2006-2-25 13:18:02.8
192.42.252.20	24	34	1,704	2,182	0	0	2006-2-25 13:18:02.8
128.104.170.17	8	4	592	280	0	0	2006-2-25 13:18:02.6
129.89.7.14	4	2	376	188	0	0	2006-2-25 13:17:56.2
128.104.230.11	2	4	188	376	0	0	2006-2-25 13:17:56.2
129.79.254.85	0	14	0	936	0	0	2006-2-25 13:18:04.6
128.104.198.26	0	4	0	296	0	0	2006-2-25 13:18:02.6
128.104.18.21	0	4	0	296	0	0	2006-2-25 13:18:02.6
129.74.4.9	0	4	0	256	0	0	2006-2-25 13:17:59.8
128.158.1.236	0	2	0	212	2	0	2006-2-25 13:18:11.1
128.158.22.1	0	1	0	154	0	0	2006-2-25 13:16:33.5
120.1.1.121	0	2	0	308	0	0	2006-2-25 13:16:59.7
128.158.1.84	0	1	0	154	0	0	2006-2-25 13:16:36.9
130.126.174.10	0	2	0	148	0	0	2006-2-25 13:16:44.8
128.158.10.94	0	1	0	110	0	0	2006-2-25 13:16:40.2
128.158.1.4	0	1	0	106	0	0	2006-2-25 13:16:43.7
128.158.1.83	0	1	0	130	0	0	2006-2-25 13:16:45.8
192.42.252.3	0	2	0	308	0	0	2006-2-25 13:17:55.7

MAC IP IPX

好，我们继续看第二个 monitor 功能，Host table，我们叫他主机列表。

这是非常好用的一个功能，有什么用呢？

第一看流量最大的 TOP10 主机，

第二看广播量有多少，当时我发现冲击波、振荡波的时候，就是看 这个 host table，发现有大量的全子网广播

第三可以快速过滤单一主机流量。

第四通过过滤功能可以看到单一业务主机的流量分布，当然也可以通过镜像接口去实现 。

我们一个一个来看。

首先 TOP10 主机，我们可以点击各列的标题来排序，方便我们分析，比如收发包情况。大家可以试一下。

第二广播量有多少

我们点击 broadcast 或 multicast 的标题，查看广播量，有一点要注意，不要忘记看 MAC 层的广播和组播，因为 MAC 的广播不一定有 IP 头，比如 ARP，同样 IP 的广播在 MAC 也可能是单播，比如子网广播。

MAC 层的广播是目的 MAC 为 48 个 1，MAC 层的组播为目的 MAC 第一个字节最低位是 1。（范老师有板书，我的本子上有，懒得画了）

IP 的广播有三种：255.255.255.255 叫本地广播，也叫直播，direct broadcast，不跨路由器。

172.16.33.255 叫子网广播，广播给 172.16.33.0 这个子网，可以跨路由器。172.16.255.255 叫全子网广播，广播给 172.16.0.0 这个主网，可以跨路由器。大家以前学网络的时候，老师会给一个概念，说路由器是三层设备，隔离广播，对吧，我也是这样给同学介绍的，但我在后面会告诉同学，并不是所有广播都隔离。

事实上只有 255.255.255.255 这类本地广播，路由器才不转发，对于子网广播和全子网广播，路由器是转发的，这是为什么呢？

我们来看 4 个 255 的广播，在 MAC 的封装中，对应的目的 MAC 是广播，而子网广播和全子网广播，对应的目的 MAC 是单播，所以路由器会转发。（范老师在演示）所以我们注意到，路由器隔离的广播是目的 MAC 为全 1 的广播，对于目的 MAC 是单播的上层广播，路由器是不能隔离的。

现在想想冲击波震荡波为什么影响那么大，因为它采用的是全子网广播，可以跨路由感染。所以对于这种流量我们要小心，希望下次再出现蠕虫病毒时，大家能快速发现，做个世界第一（同学们笑），同样我们要关注 MAC 层的广播。第三，就是我们可以关注单一主机流量。

第一种办法，抓包。选中主机，点一下抓蝴蝶的工具，这样通过专家系统和解码你就可以分析他在干什么了。这个我们后面再讲 。

第二种办法，用 single station。选中主机，点一下下面这个电脑的图标，你可以看到他在跟谁通信，如果你看到他跟几十台、上百台机器同时通讯，可能是什么？（同学们：BT)，对，像 BT,电驴等 P2P 应用会有这个特征。

第四，就是我们如果我们把单一业务服务器的接口镜像过来，我们就可以看到这台机器的流量状况，我们也可以采用过滤的方式。

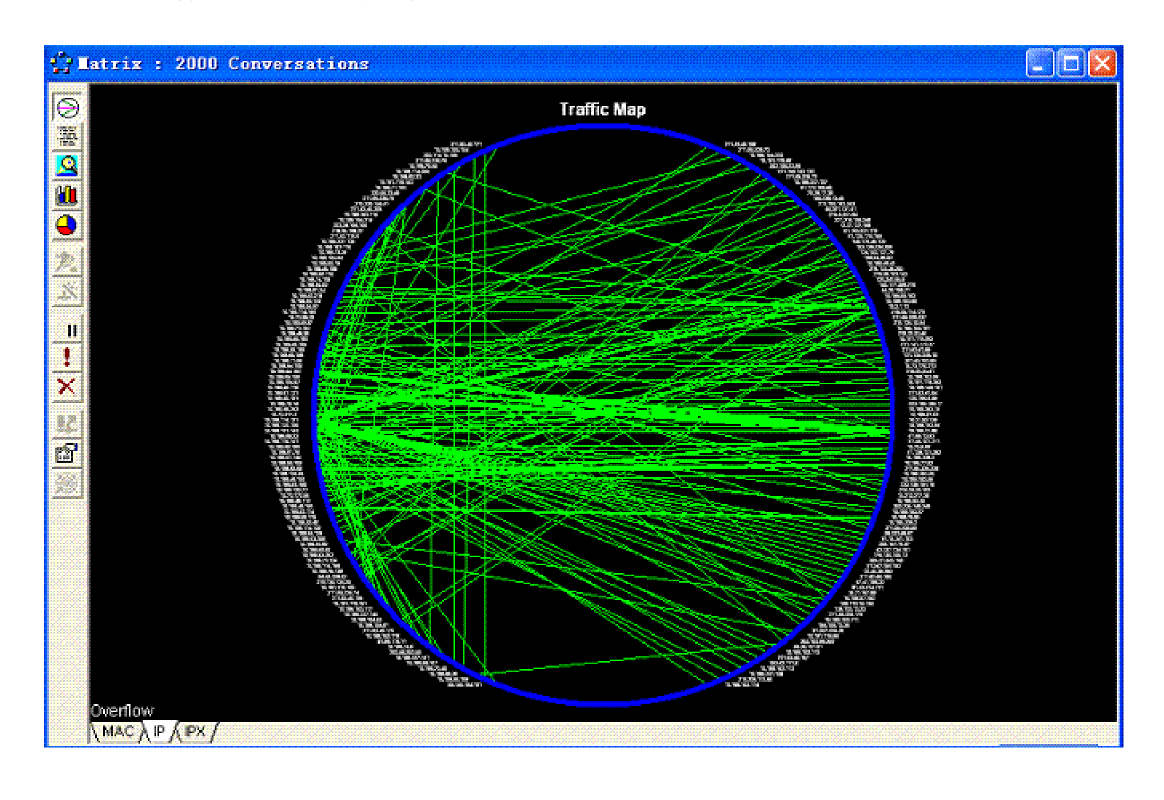
Sniffer 有一种叫 Monitor 过滤器。大家选中一台机器，假设这是你要关心的业务主机，再点一下这个定义过滤器的图标，（范老师在演示），你看他自动产生一个叫 NEW1 的过滤器，就是这台机器跟任何机器通讯这样的一个过滤器。我们点一下确定。

我们在选择 monitor 菜单上的 select filter，选 apply monitor filter，再选 new1，确定。

大家注意到，现在 host table 就只有和这台机器通讯的所有主机流量情况。要注意一点是，monitor filter 应用的时候，对所有 monitor 功能生效，所以在分析单一业务的时候，特别好用。当然如果你们买的是 InfiniStream 的话，就更方便了，想分析那个业务就分析哪个业务。

怎么样？Host table 好用吧？（同学问：为什么广播也是一台主机？不是说广播地址不会作为主机地址吗？）（编者：这个问题好像比较低级）。

这是流量分析技术的特点，再流量分析中，它纯粹从包结构中去取得主机信息，也就是目的 MAC,源 MAC,目的 IP，源 IP,他都作为主机处理，广播地址不会在原地址中出现，但在目的地址中出现，也是一台主机。这并不影响我们分析。好。还有什么问题吗？大家用 5 分钟自己试一下。

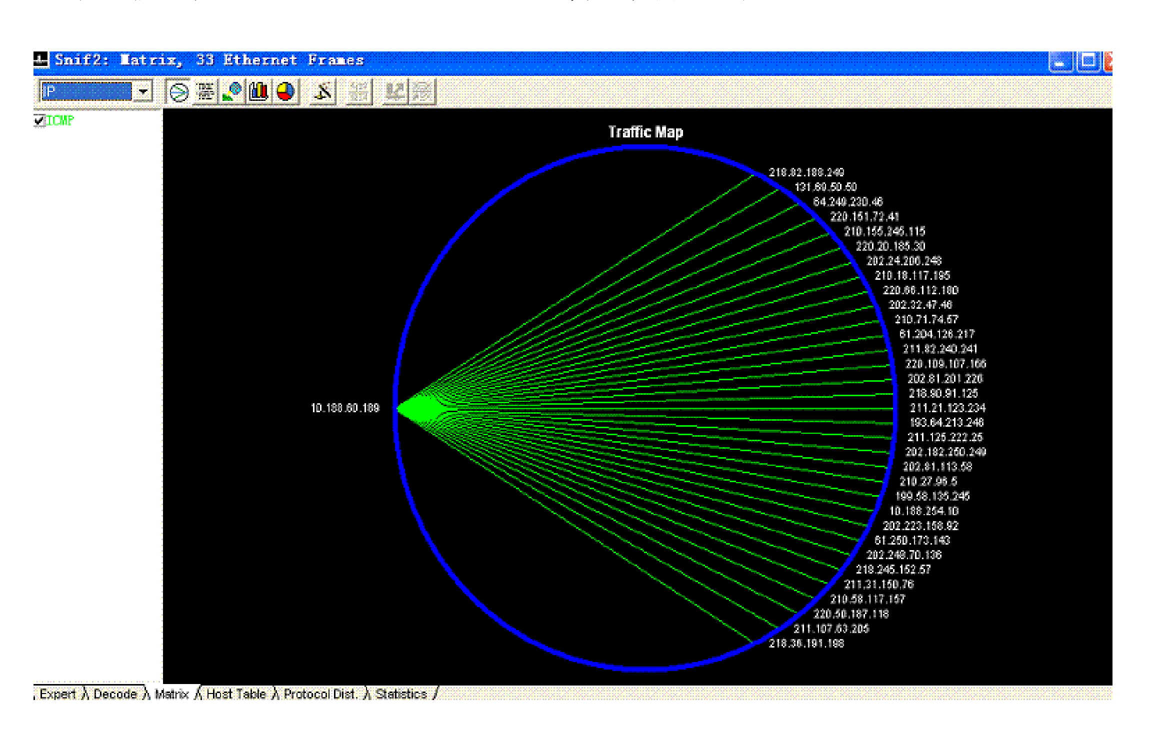


好，我们继续看第 3 个 monitor 功能，Matrix，我们叫他矩阵，其实就是主机会话情况，很多人用他来发现病毒，其实用他来评估网络状况，和异常流量，是一个很好用的工具。

大家看，一下子就满了，大多数网络中都是这样的，我们可以按一下崭停。然后来分析

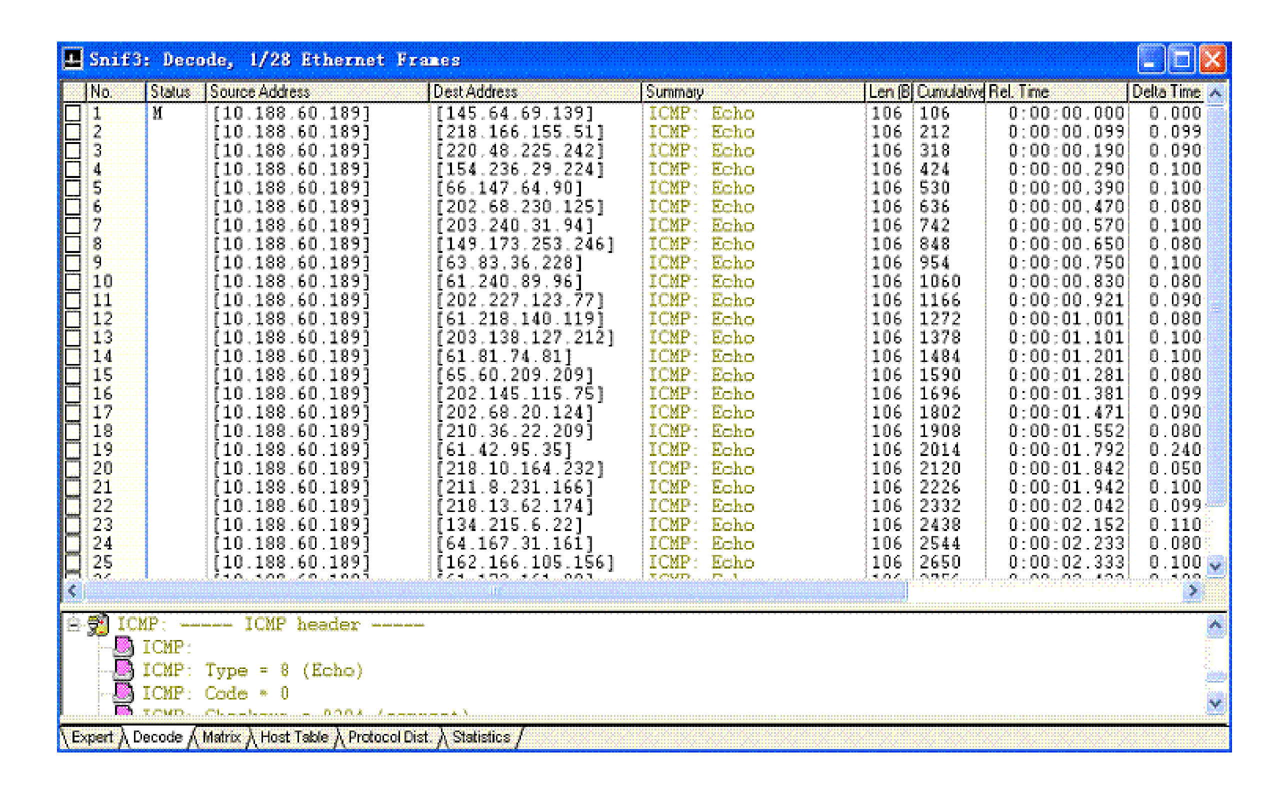
分析什么呢？

看那一台主机的连接数最多，要注意这个连接数不是传输层的连接数，是指谁跟最多的主机连接 ，按右建选 zoom，放大。 ，找到对外连接最多的机器，选中，按右建，选 show select nodes,大家自己试一下。



我们注意到这台机器跟很多机器通讯，这正常吗？（同学们：不正常）这要看实际情况，如果这时一台业务主机，太正常了，如果这时一台 PC 机，或许在作 P2P。

究竟在作什么呢?


Snif3: Decode, 1/28 Ethernet Frames

No.	Status	Source Address	Dest Address	Summary	Len (B)	Cumulative	Rel. Time	Delta Time
1	M	[10.188.60.189]	[145.64.69.139]	ICMP: Echo	106	106	0:00:00.000	0.000
2		[10.188.60.189]	[218.166.155.51]	ICMP: Echo	106	212	0:00:00.099	0.099
3		[10.188.60.189]	[220.48.225.242]	ICMP: Echo	106	318	0:00:00.190	0.090
4		[10.188.60.189]	[154.236.29.224]	ICMP: Echo	106	424	0:00:00.290	0.100
5		[10.188.60.189]	[66.147.64.90]	ICMP: Echo	106	530	0:00:00.390	0.100
6		[10.188.60.189]	[202.68.230.125]	ICMP: Echo	106	636	0:00:00.470	0.080
7		[10.188.60.189]	[203.240.31.94]	ICMP: Echo	106	742	0:00:00.570	0.100
8		[10.188.60.189]	[149.173.253.246]	ICMP: Echo	106	848	0:00:00.650	0.080
9		[10.188.60.189]	[63.83.36.228]	ICMP: Echo	106	954	0:00:00.750	0.100
10		[10.188.60.189]	[61.240.89.96]	ICMP: Echo	106	1060	0:00:00.830	0.080
11		[10.188.60.189]	[202.227.123.77]	ICMP: Echo	106	1166	0:00:00.921	0.090
12		[10.188.60.189]	[61.218.140.119]	ICMP: Echo	106	1272	0:00:01.001	0.080
13		[10.188.60.189]	[203.138.127.212]	ICMP: Echo	106	1378	0:00:01.101	0.100
14		[10.188.60.189]	[61.81.74.81]	ICMP: Echo	106	1484	0:00:01.201	0.100
15		[10.188.60.189]	[65.60.209.209]	ICMP: Echo	106	1590	0:00:01.281	0.080
16		[10.188.60.189]	[202.145.115.75]	ICMP: Echo	106	1696	0:00:01.381	0.099
17		[10.188.60.189]	[202.68.20.124]	ICMP: Echo	106	1802	0:00:01.471	0.090
18		[10.188.60.189]	[210.36.22.209]	ICMP: Echo	106	1908	0:00:01.552	0.080
19		[10.188.60.189]	[61.42.95.35]	ICMP: Echo	106	2014	0:00:01.792	0.240
20		[10.188.60.189]	[218.10.164.232]	ICMP: Echo	106	2120	0:00:01.842	0.050
21		[10.188.60.189]	[211.8.231.166]	ICMP: Echo	106	2226	0:00:01.942	0.100
22		[10.188.60.189]	[218.13.62.174]	ICMP: Echo	106	2332	0:00:02.042	0.099
23		[10.188.60.189]	[134.215.6.22]	ICMP: Echo	106	2438	0:00:02.152	0.110
24		[10.188.60.189]	[64.167.31.161]	ICMP: Echo	106	2544	0:00:02.233	0.080
25		[10.188.60.189]	[162.166.105.156]	ICMP: Echo	106	2650	0:00:02.333	0.100

我们注意到这台机器向公网发出大量的 ICMP 包，那是在作什么？（同学们：在 ping）对！PING 采用 ICMP 协议，ping 可以用来扫描，也可以用来攻击。扫描就是看那一台机器活着，接着扫描端口，在攻击，所以扫描是攻击主机的前奏。

另外 ，还可以用 ping 来冲击路由器，或占用带宽，是一种 DOS 攻击。大家看这个过程更像哪一种类型。

（同学们：扫描，DOS 攻击）

一般情况下，扫描会是比较连续的地址，我们看这个地址并不连续，我们先排除扫描，当然不是绝对的，也有比较聪明的扫描。

有同学说，这是 DOS 攻击，那是冲击路由器，还是占用带宽？

（同学们：冲击路由器）

嘿，这次比较统一，我也觉得他在冲击路由器，我们看，他的目标地址基本不在一个网段，这样路由器收到这样的数据包会消耗大量资源在查找路由表上面。所以对路由器有一定冲击。

一般来说，如果他想占用带宽的话，会发大包，我们发现，包的长度不大，并且一秒钟才发 10 几个包，所以对贷款冲击不大。

或许大家会觉得这没秒 10 几个包对路由器冲击也不大呀。大家想像一下，如果有很多机器在作这个操作，那影响就会很大。

大家自己在找一找，是否还有其他机器在作同类事情。

（同学们找出 7 台这样的机器）

好大家找出 7 台这样的机器，怎么找出来的？有同学用钢材的办法，有同学用过滤，都市好办法。

现在假设在你们的网络中出现这样的情况，我们发现了异常，接下来怎么做？

（同学们：找到这台机器）

然后呢？

我们可以看看这台机器的任务管理器，看看有什么不常见的进程，把他去掉，看是否解决。在看其他的机器，是否有类似的特征。
这是我的一个学员发给我的，当时他发现这 7 台机器都有一个特殊的进程，但是他的防病毒软件没有查出来。他手工解决了。

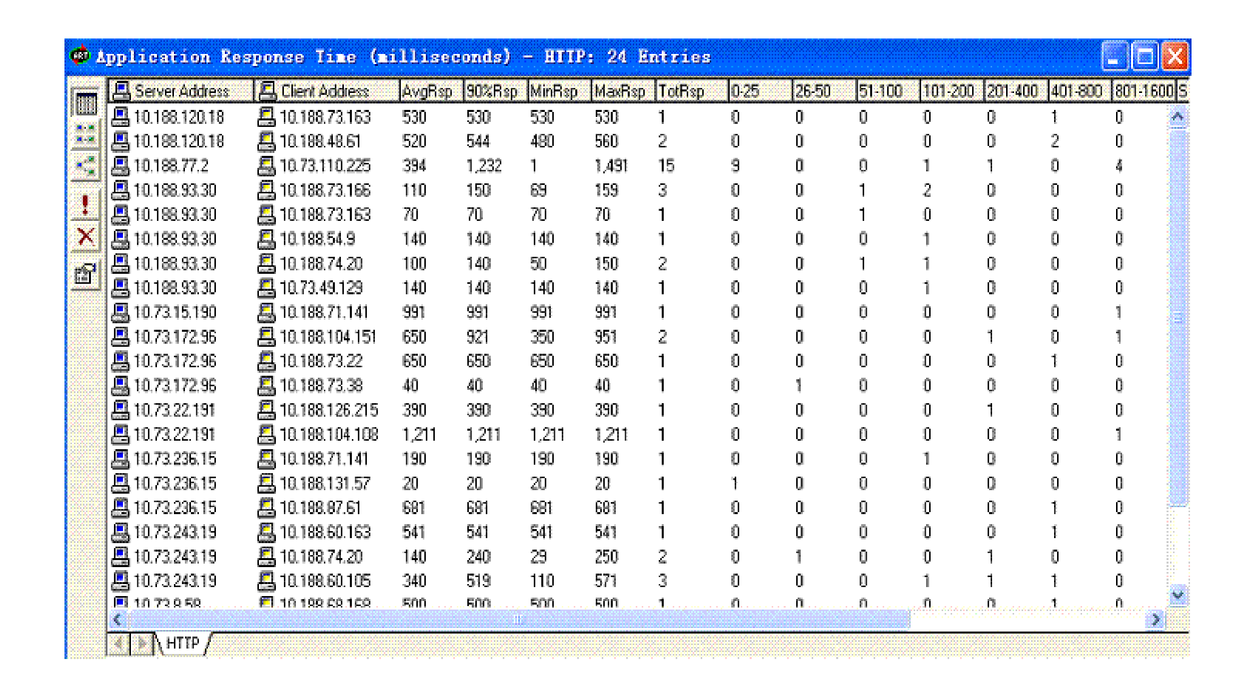
这很好说明用 Sniffer 可以比防病毒软件更快发现病毒，因为防病毒软件是后知后觉得，什么意思？防病毒软件必须有相应的特征才能查病毒。而 Sniffer 通过流量可以发现一些特征，一些异常。

但是有一点，我们不能拿 Sniffer 当防病毒软件用，那不是他的特长，同时也太低沽 Sniffer 的功能了（同学们笑）

好我们在看看扫描是怎么一回事，大家看这个 trace file（范老师在演示，我就不写了）

先是 ARP 扫描，再端口扫描，接下来就是攻击了。

（编者：接着我们做了一个游戏，范老师让大家用 Sniffer 攻击他的机器，结果 1 台机器就把他的机器搞死了，这个就不细说了）



Application Response Time (milliseconds) - HTTP: 24 Entries

Server Address	Client Address	AvgRsp	90%Rsp	MinRsp	MaxRsp	TotRsp	0-25	26-50	51-100	101-200	201-400	401-800	801-1600
10.188.120.18	10.188.73.163	530	530	530	530	1	0	0	0	0	0	1	0
10.188.120.18	10.188.48.61	520	544	480	560	2	0	0	0	0	0	2	0
10.188.77.2	10.73.110.225	394	1,232	1	1,491	15	9	0	0	1	1	0	4
10.188.93.30	10.188.73.166	110	150	69	159	3	0	0	1	2	0	0	0
10.188.93.30	10.188.73.163	70	70	70	70	1	0	0	1	0	0	0	0
10.188.93.30	10.188.54.9	140	140	140	140	1	0	0	0	1	0	0	0
10.188.93.30	10.188.74.20	100	140	50	150	2	0	0	1	1	0	0	0
10.188.93.30	10.73.49.129	140	140	140	140	1	0	0	0	1	0	0	0
10.73.15.190	10.188.71.141	991	991	991	991	1	0	0	0	0	0	0	1
10.73.172.96	10.188.104.151	650	921	350	951	2	0	0	0	0	1	0	1
10.73.172.96	10.188.73.22	650	650	650	650	1	0	0	0	0	0	1	0
10.73.172.96	10.188.73.38	40	40	40	40	1	0	1	0	0	0	0	0
10.73.22.191	10.188.126.215	390	390	390	390	1	0	0	0	0	1	0	0
10.73.22.191	10.188.104.108	1,211	1,211	1,211	1,211	1	0	0	0	0	0	0	1
10.73.236.15	10.188.71.141	190	190	190	190	1	0	0	0	1	0	0	0
10.73.236.15	10.188.131.57	20	20	20	20	1	1	0	0	0	0	0	0
10.73.236.15	10.188.87.61	681	681	681	681	1	0	0	0	0	0	1	0
10.73.243.19	10.188.60.163	541	541	541	541	1	0	0	0	0	0	1	0
10.73.243.19	10.188.74.20	140	240	29	250	2	0	1	0	0	1	0	0
10.73.243.19	10.188.60.105	340	519	110	571	3	0	0	0	1	1	1	0

HTTP

好，我们再看第四个 monitor 功能，ART，Application Response Time，应用响应时间。

应用响应时间是分析应用的一个很好工具，主要用来分析应用的性能。ART 是指一个客户端发出一个请求，到服务器响应回来的时间差。

一般来说，应用响应的快慢，是应用性能的一个重要指标。

应用性能主要决定于几个因素：网络因素、服务器因素、客户端因素、应用协议因素 。

我们先看看如何操作,再来看看应用这个功能。

我们打开 ART,大家看到 Http 的应用响应时间分析,这里有几个列,server Address,Client Address.

他是怎么知道谁是 Server，谁是 Client?其实也就是看端口号和 IP 的对应关系，比如如果一个数据包的目的 IP 是 1.1.1.1，目的端口是 80，Sniffer 就会认为 1.1.1.1 就是 Http 服务器。对应的源 IP 就是 Client。

AvgRsp—平均响应时间

90%Rsp—90%响应时间，去掉头尾个 5%，其实我个人觉得去掉最大的 10%更合理一些。

还有最大最小的响应时间，这些都是以毫秒为单位。

接着就是 TotalRsp,这个是响应次数，单位是次。

接着是 0 到 25 毫秒的响应有多少次，25 到 50 毫秒的响应有多少次。。等等。

后面还有 server 发送子节数，client 发送子节数，timeout 次数等等，5 秒不响应则 Timeout。

我们再看看怎么增加其他应用，按属性，选择 display protocol,添加你关心的协议，再确定，ART 会重新刷新（范老师在演示）
你看我这里就有了 telnet,Oracle。

（同学们：我们没有 Oracle)

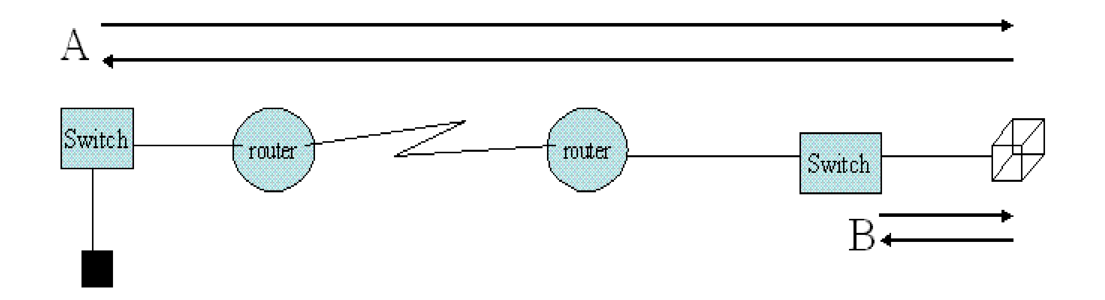
我知道，其实平时我们更关心的是我们关键业务，所以我们要把我们关键业务的端口添加进来，怎么添加？大家跟我来，选菜单上的“tool”->“option”“protocol”，拉到下面,添加一种应用，比如 Oracle,端口 1521。

再在属性里把这个新协议选上，有了吗？（跑去解答问题去了）

好，大家都做出来了，我们平常分析关键业务就行了，有一点要说明，一种业务可能有多个应用，也就是多个端口，需要同时分析。

有些同学喜欢把所有 well known 的协议添加到协议列表里，我在共享目录上有两个注册表注入工具，大家只要运行以下就可以将这些常用端口都注入到协议列表里，就不用一个一个敲了。其实我个人觉得不太必要，多了反而乱。
大家打开注册表，我们看一下协议列表，找到这两项：
HKEY_LOCAL_MACHINE\SOFTWARE\Network Associates, Inc.\Sniffer\4.7\1CommonSettings\Protocols\IP Protocols\TCP
HKEY_LOCAL_MACHINE\SOFTWARE\Network Associates, Inc.\Sniffer\4.7\1CommonSettings\Protocols\IP Protocols\UDP
这就是协议列表。注意不要有重复的，否则会报错。

ART



（编者：这是范老师的板书）

应用响应时间是评估影响应用性能因素的一种很好的工具。我们看这样一个例子。

比如通过 client 通过广域网连接到服务器。

我们同时在 AB 两点部署 Sniffer，分析某一业务的响应时间。

假设 Sniffer 在 A 点，他所看到的响应时间包括网络消耗时间和服务器处理时间在 B 点的 Sniffer 看到的响应时间主要是服务器处理时间。这样我们比较 AB 两点的响应时间，来判断影响性能的主要因素是网络还是服务器。

假设 A 点的响应时间是 400 毫秒，B 点的响应时间是 100 毫秒，我们就知道 A 点的 400 毫秒中有 300 毫秒是消耗在网络上的，我们可以认为对于这个业务，性能的主要瓶颈在网络上，如果我们在深入分析是距离因素还是贷款因素，我们就可以判断是否有改善空间。这个细节我们在第二门课讲。
如果 B 点的响应时间达到 250 毫秒，我们可以认为改善服务器的性能对于这个应用来说会更明显一些。

如果我们 Sniffer 用多了。我们就可以做一个 AB 点的响应时间的基准线，假设正常情况下 A 点的响应时间是 400 毫秒，有一天你发现平均响应时间达到 600 毫秒，你就应该关注了，或许用户还没有抱怨，如果你这时分析应用性能下降的原因，你就可以避免故障的产生，同时避免用户投诉。当然你也会有 B 点的基准线，比较跟平时有何不同，很快就知道应该检查网络还是服务器。

对于 ART 还有什么不清楚地吗？或者大家平常还有其他用法？

（编者：本来这次想写完第一天课程再共享，看到论坛中大家在催，就先写到这吧，过几天就可以把第一天全部内容发完）

（编者：范老师已经找过我了，他说他的 MSN 有很多陌生人，后来看了内容就知道是我写的，他说分享知识可以，但不要透露敏感信息，还说我比他讲得好，汗。。。。所以这次我没有完全按他的录音翻译，自己有所删节，但基本上是原汁原味。）

（编者：我发现写这个真的很累，因为有很多演示，很难写出来，最近很忙，我都没有忘记我的承诺，给我加油，好吗？）

（编者：所有 PPT 都市我自己做的，范老师的 ppt 是不给我们的，我很努力吧！）

自已的一些看法

看完《范伟导老师 sniffer 课程资料》大家有什么想法，大家可以提出来讨论，当然也可以跟身边有共同爱好朋友一起讨论，让大家讨论的目的就是让大家有一个思想交流的过程，我不知道大家对这个是什么想法，我的想法就跟当初学路由器，交换机和 TCP/IP 一样，突然觉得搞网络的不懂这个东西就枉然了，跟路由器差不多，路由器，交换机是网络的的核心东西，你不搞懂这个你就白学网络了，我想学 sniffer 也一样，你不懂 sniffer，你也枉然了，你想，在工作中你不能很好的排除网络中存在的故障，不能很好的发现网络中存在的问题，不能跟老板解释清楚故障所在，那么从根本上你就是不称职的，是吧，所以学习 sniffer 不但是对自已的一个提高，更是对自已的负责，我们搞技术的不能为了几个工钱什么都干不了啊，呵，说的太严肃了点。

不知范老师的课程大家看了几次，呵，这篇稿子中的有些图是我自已添上去的怕大家不明白，其中有些出入还请大家见谅。使用时有一个问题我不得不提出来，在 windows2003 中运行 Dashboard 表可能会出现下面这个问题：



所以最好在 win2000 下运行，大家有时间去查查，我的学习时间快结束了，就来不及了，所以就此打住，有时间再向范老师请教。

我相信在上一篇范老师的课程记录中，大家应该明白了关于 sniffer 的一些基本应用，当然其中说的网络技术人员的职业发展也应对大家有所启示才对，还希望大家能明白，不是很费力才对，网络这东西有的时候说需要自已去琢磨，单纯的靠理论知识也解决不了问题，因此建议大家在学习过程中自已的电脑上一

定要安装上 sniffer 才好，不能凭空的在这听我说，那就没有意义了，也枉费我花心思去整理这么一片东西，希望对大家有所帮助，要告诉大家的是我用的是 SnifferPro_4_70_530 这个版本的，当然现在也有更高版本的，大家可以在网上找的到，但可能在安装时要注册所以请大家在安装前务必先准备一个能用的注册码，好了，费话就不说了，先谈到这里，以后有什么内容再慢慢来。

在前面的那一稿中主要提到了几点，一是 sniffer 的作用，其中说到一点就是检测病毒，当然这个病毒是指蠕虫病毒，因为这个病毒发作时会对网络产生很大的影响，首先就是网络的数据流量会加大很多，这个就可以做为根据去判断发生源，这对网管来说是一个很好的东西，能够发现问题嘛！当然还有很多作用如：评估网络的性能，快速定位故障，排除潜在的威胁，排除来自内部的威胁。就这些对网络管理都很有用啊，这就是工具给我们带来的方便啊，你自已说说，是这样不，对于网络管理来说，这些就是我们最基本的工作啊。你说一个东西能帮我们解决这么多的问题那我们还找什么？

第二点就是基本介绍的了 sniffer 的一些基本功能及界面和一些使用，这方便我们在这里会详细的介绍，如果大家不是很熟的话，我们也可以把安装过程给大家讲解一下，这个我可以通过抓图的方式让大家很明白这个过程，当然在最后我们还会通过一些实例来讲解它的使用及如果对一些常用的功能进行使用，不过请大家不要拿来做别的事，那就脱离我写这些东西的本意了。再次重申一下，我写这些东西都是为了让大家明白一个工具的使用及原理让大家在工作能更好的完成自已的任务，因此请大家不要用来做为其它的用途。

第三点就是关于一些协议的东西，学习 sniffer 建议最好学过 CCNP，当然你有网络基础也是一样的，这里就不多说了，在后面我们也会着重讲解这方面的知识，所以希望大家不要着急，在 sniffer 中 TCP/IP 协议是很重要的，因为在使用过程中我们都要对 TCP/IP 协议进行分析。

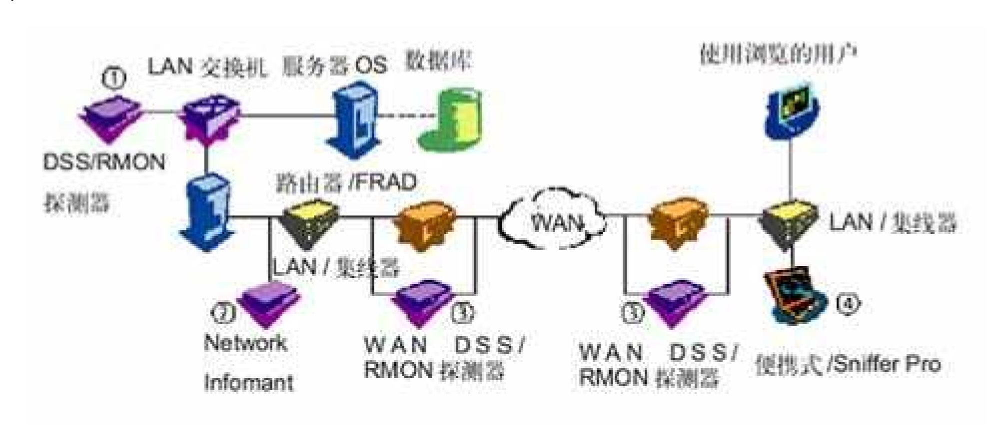
好了不多说了，说的再多还不如让大家自已动手，好吧，开始我们的学习过程。

NAI 的 Sniffer 系统

网络对你公司的作用愈来愈重要，同时随之而来的是你要为网络故障付出越来越昂贵的代价。网络速度变慢和停机可以轻而易举使你损失数十万美元的收入。为确保客户和合作伙伴的网络正常使用，在瓶颈造成故障之前，你应该应用网络安全检测和解决的方案。

确保网络的稳定性

在电子商务时代，企业网络已经触及到你公司的每一个角落。在全球范围内，每时每地都有员工使用网络访问商业应用程序，每时每地都有客户登录网站进行交易。为确保持续的网络性能，你需要拥有先进的监控和故障解决工具。Sniffer 作为减少网络故障的解决方案，这个先进的软件包提供的网络管理工具，可以帮助你主动监控网络，在故障对使用者、客户和你公司的基础线路产生影响之前将其解决。



Sniffer Technologies

适用于当今复杂的企业网络

Sniffer 能确保整个 LAN 和 WAN 拓扑网络的最高性能，从 10/100 兆以太网到最新的高速 ATM 和千兆位主干网-----贯穿所有企业和 Internet。

可以对整个网络的所有七层进行分析

Sniffer 软件可以对所有网络七层进行主动的监控和故障解决----从物理层到应用层----所有这些功能都可以实时实现。

确保网络性能

Sniffer Informant 软件将为你提供关于网络状况和性能的完整描述，从可使用的带宽网络利用率到应用程序的效率。

可进行完整的网络监测

为使你的网络运行能够保持最高的效率，Sniffer Technologies 提供了一整套的便携式及分布式软件工具，可以对网络性能进行监控、故障解决、报告和主动管理。它是用于企业网络故障和性能管理的智能工具系列。

显然，没有两个网络具有相同的配置、用户基准、需求或设备。Sniffer 在集成的软件包中提供全套主要组件，可以满足你对网络的精确要求。

Sniffer 便携式分析软件包

实时网络分析

如果要迅速检测和解决网络故障和性能问题，NAI 的便携式 Sniffer 的专家分析功能可以找出网络、数据库和应用程序故障的根本原因。

Sniffer 的网络分析器可以运行于桌面、便携式和笔记本 PC，使用了 400 多种协议解释和强大的专家分析功能，可以对网络传输进行分析，找出故障和响应缓慢的原因。它甚至可以对多拓扑、多协议网络进行分析---所有这些功能都可以自动地实时实现。

部门网采用 Sniffer Basic，可以使用 Sniffer 的监控和解释分析功能。低成本的实时监控和解释功能使得 Sniffer Basic 成为 MIS 人员的理想选择，他们可以对小型企业、远程办公室和部门网进行支持。同时，Sniffer Basic 为一线的 IS 人员提供了一个可以进行常规故障解决的强大工具。

Sniffer Pro LAN 和 Sniffer Pro WAN

它们适用于有完整的专家分析和 Sniffer 先进的协议解释功能要求的网络。Sniffer Pro 对 LAN 和 WAN 网段上的网络传输的所有层进行监测，揭示性能问题，分析反常情况，并推荐解决方案---所有这些功能都可以自动地实时实现。

Sniffer Pro High-Speed

它是用于优化最新的 ATM 及千兆位以太网性能和其可靠性的工具，Sniffer 的 SmartCapture 功能提供了对 LAN 仿真数据流和 IP 交换环境的监测。

支持 ATM，它在企业网络中，针对每个 ATM OC-3 和 OC-12 高速链路，为你提供了一个单独的解决方案。

支持 Gigabit，它是以全双工速度捕获，解决千兆位协同性问题的分析器。其强大的处理功能可以提供对千兆位以太网的非对称监测。

Sniffer 分布式分析软件包

集成的专家分析和 RMON 监控软件包。

NAI 的分布式 Sniffer 解决方案对应用程序传输和网络设备状态进行全天候分析，可以使应用程序保持最高的运行效率。其中包括对部门网、校园网和主干网的集中监控、设备级别报告和故障解决。

分布式 Sniffer 系统

从一个单独的管理控制台启动自动 RMON-适应网段监控和故障识别。Sniffer 专家分析软件可以使你以最快的速度解决故障---即使有复杂的网络拓扑、协议和应用程序。

企业故障和网络性能管理解决方案--Distributed Sniffer System/RMON 可以对整个网络中的主要网段（LAN、WAN、ATM 和千兆位以太网）提供网络监控、协议解释和专家分析功能。基于标准的监控和专家分析的强有力的结合使之成为多拓扑结构和多协议网络的最优管理工具。

Network Informant Suite

对总体网络性能作出报告

NAI 强大的基于浏览器的网络信息解决方案可以帮助你识别趋势并管理服务等级。这个企业报告解决方案采取了主动的方式进行网络管理。它通过网络自动从设备中获取数据，然后编译成为网络性能的图形视图。基于浏览器的报告可以提供总体概要分析和详细异常分析，帮助你管理服务级别，控制 WAN 线路占用，减少网络故障。

Network Informant 是用于企业网络性能报告、服务等级管理、WAN 线路占用控制和企业诊断的首要的高级解决方案。灵活的基于浏览器的解决方案包括标准的和任选的报告，可以进行配置以满足你公司的特殊需求。Network Informant

还可以允许你使用 Crystal Reports 创建自定义报告，即业界公认的关系数据库报告系统。

RouterPM 也是 Network Informant 软件包中的一个集成部分，为你提供了进行企业 WAN 和 LAN 连续性能分析的的工具。它具有预处理和异常报告功能，使你可以预先发现问题并避免故障。

Sniffer Predictor

当网络需要进行升级时，Sniffer Predictor 将进行预报，确保资源适当以保持运行通畅。

Sniffer Predictor 是一个基于工具的解决方案，它可以在几秒内预测变化将对网络性能产生何种影响。适用于 Sniffer 便携式分析软件包和 Sniffer

分布式分析软件包。Sniffer Predictor 将供给企业网络管理员和计划者日常使用，他们需要的整个网络及时的性能及计划数据。

这里对 NAI 的 Sniffer pro 进行介绍了。让大家大至上了解一下，在后面的章节里我们主要对 Sniffer pro 进行介绍。

嗅探的基本原理

一 前言

SNIFF 真是一个古老的话题，关于在网络上采用 SNIFF 来获取敏感信息已经不是什么新鲜事，也不乏很多成功的案例，那么，SNIFF 究竟是什么呢？SNIFF 就是嗅探器，就是窃听器，SNIFF 静悄悄的工作在网络的底层，把你的秘密全部记录下来。看过威尔史密斯演的《全民公敌》吗？SNIFF 就象里面精巧的窃听器一样，让你防不胜防。

SNIFF 可以是软件，也可以是硬件，既然是软件那就要分平台，有 WINDOWS 下的、UNXI 下的等，硬件的 SNIFF 称为网络分析仪，反正不管硬件软件，目标只有一个，就是获取在网络上传输的各种信息。本文仅仅介绍软件的 SNIFF。

当你舒适的坐在家里，惬意的享受网络给你带来的便利，收取你的 EMAIL，购买你喜欢的物品的时候，你是否会想到你的朋友给你的信件，你的信用卡帐号变成了一个又一个的信息包在网络上不停的传送着，你是否曾经这些信息包会通过网络流入别人的机器呢？你的担忧不是没有道理的，因为 SNIFF 可以让你的担忧变成实实在在的危险。就好象一个人躲在你身后偷看一样………

二 网络基础知识

“网络基础知识”，是不是听起来有点跑题了？虽然听起来这和我们要谈的 SNIFF 没什么关系，可是还是要说一说的，万丈高楼平地起，如果连地基都没打好，怎么盖楼？！如果你对网络还不是十分清楚的话，最好能静下心来好好看看，要知道，这是基础的基础，在这里我只是简单的说一下，免得到时候有人迷糊，详细的最好能够自己去找书看看。

（1） TCP/IP 体系结构

开放系统互连（OSI）模型将网络划分为七层模型，分别用以在各层上实现不同的功能，这七层分别为：应用层、表示层、会话层、传输层、网络层、数据链路层及物理层。而 TCP/IP 体系也同样遵循这七层标准，只不过在某些 OSI 功能上进行了压缩，将表示层及会话层合并入应用层中，所以实际上我们打交道的 TCP/IP 仅仅有 5 层而已，网络上的分层结构决定了在各层上的协议分布及功能实现，从而决定了各层上网络设备的使用。实际上很多成功的系统都是基于 OSI 模型的，如：如帧中继、ATM、ISDN 等。

TCP/IP 的网络体系结构（部分）

```
-----------------------------------
| SMTP | DNS | HTTP | FTP | TELNET| 应用层
-----------------------------------
|     TCP      |     UDP          |   传输层
-----------------------------------
|  IP  |    ICMP     | ARP RARP | 网络层
------------------------
| IEEE 802 以太网 SLIP/PPP PDN etc| 数据链路层
-----------------------------------
|       网卡 电缆 双绞线 etc      | 物理层
-----------------------------------
```

从上面的图中我们可以看出，第一层物理层和第二层数据链路层是 TCP/IP 的基础，而 TCP/IP 本身并不十分关心低层，因为处在数据链路层的网络设备驱动程序将上层的协议和实际的物理接口隔离开来。网络设备驱动程序位于介质访问子层(MAC)

（2） 网络上的设备

中继器：中继器的主要功能是终结一个网段的信号并在另一个网段再生该信号，一句话，就是简单的放大而已，工作在物理层上。

网 桥：网桥使用 MAC 物理地址实现中继功能，可以用来分隔网段或连接部分异种网络，工作在数据链路层。

路由器：路由器使用网络层地址(IP,X.121,E.164 等)，主要负责数据包的路由寻径，也能处理物理层和数据链路层上的工作。

网 关：主要工作在网络第四层以上，主要实现收敛功能及协议转换，不过很多时候网关都被用来描述任何网络互连设备。

（3）TCP/IP 与以太网

以太网和 TCP/IP 可以说是相互相成的，可以说两者的关系几乎是密不可分，以太网在一二层提供物理上的连线，而 TCP/IP 工作在上层，使用 32 位的 IP 地址，以太网则使用 48 位的 MAC 地址，两者间使用 ARP 和 RARP 协议进行相互转换。从我们上面 TCP/IP 的模型图中可以清楚的看到两者的关系。

载波监听/冲突检测(CSMA/CD)技术被普遍的使用在以太网中，所谓载波监听是指在以太网中的每个站点都具有同等的权利，在传输自己的数据时，首先监听信道是否空闲，如果空闲，就传输自己的数据，如果信道被占用，就等待信道空闲。而冲突检测则是为了防止发生两个站点同时监测到网络没有被使用时而产生冲突。以太网采用广播机制，所有与网络连接的工作站都可以看到网络上传递的

数据。

为了加深你的理解，我们来看看下面的图，一个典型的在以太网中客户与服务器使用 TCP/IP 协议的通信



以太网

唆唆了这么多，有人烦了吧？相信我，这是基础的基础，可以说是说得是很简单拉，如果需要，拿出个几十万字来说上面的内容，我想也不嫌多，好了，让我们进入下一节，sniff 的原理。

三 SNIFF 的原理

要知道在以太网中，所有的通讯都是广播的，也就是说通常在同一个网段的所有网络接口都可以访问在物理媒体上传输的所有数据，而每一个网络接口都有一个唯一的硬件地址，这个硬件地址也就是网卡的 MAC 地址，大多数系统使用 48 比特的地址，这个地址用来表示网络中的每一个设备，一般来说每一块网卡上的 MAC 地址都是不同的，每个网卡厂家得到一段地址，然后用这段地址分配给其生产的每个网卡一个地址。在硬件地址和 IP 地址间使用 ARP 和 RARP 协议进行相互转换。

在正常的情况下，一个网络接口应该只响应这样的两种数据帧：

1.与自己硬件地址相匹配的数据帧。

2.发向所有机器的广播数据帧。

在一个实际的系统中，数据的收发是由网卡来完成的，网卡接收到传输来的数据，网卡内的单片程序接收数据帧的目的 MAC 地址，根据计算机上的网卡驱动程序设置的接收模式判断该不该接收，认为该接收就接收后产生中断信号通知 CPU，认为不该接收就丢掉不管，所以不该接收的数据网卡就截断了，计算机根本就不知道。CPU 得到中断信号产生中断，操作系统就根据网卡的驱动程序设置的网卡中断程序地址调用驱动程序接收数据，驱动程序接收数据后放入信号堆栈

让操作系统处理。而对于网卡来说一般有四种接收模式：

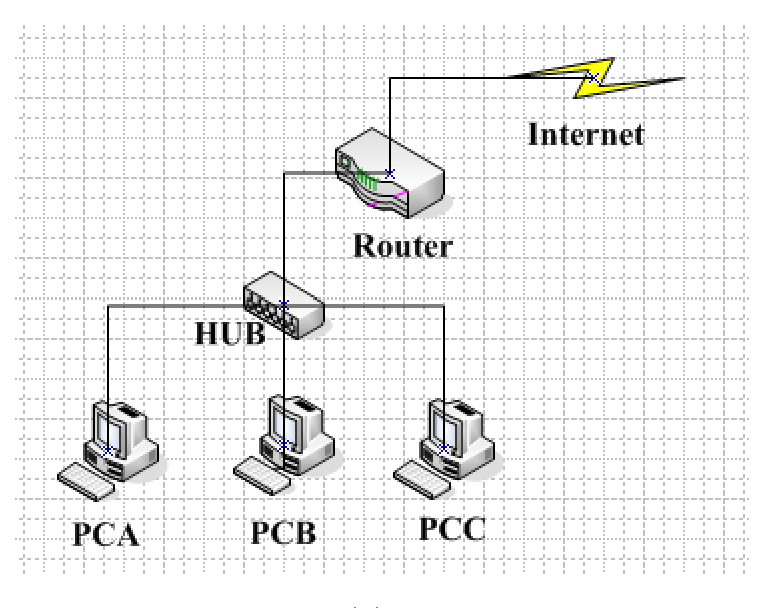
广播方式：该模式下的网卡能够接收网络中的广播信息。

组播方式：设置在该模式下的网卡能够接收组播数据。

直接方式：在这种模式下，只有目的网卡才能接收该数据。

混杂模式：在这种模式下的网卡能够接收一切通过它的数据，而不管该数据是否是传给它的。

好了，现在我们总结一下，首先，我们知道了在以太网中是基于广播方式传送数据的，也就是说，所有的物理信号都要经过我的机器，再次，网卡可以置于一种模式叫混杂模式（promiscuous)，在这种模式下工作的网卡能够接收到一切通过它的数据，而不管实际上数据的目的地址是不是他。这实际上就是我们SNIFF工作的基本原理：让网卡接收一切他所能接收的数据。



（图一）

我们来看一个简单的例子，如图一所示，机器A、B、C与集线器HUB相连接，集线器HUB通过路由器Router访问外部网络。这是一个很简单也很常见的情况，比如说在公司大楼里，我所在的网络部办公室里的几台机器通过集线器连接，而网络部、开发部、市场部也是同样如此，几个部门的集线器通过路由器连接。还是回到我们的图一上来，值得注意的一点是机器A、B、C使用一个普通的HUB连接的，不是用SWITCH，也不是用ROUTER，使用SWITCH和ROUTER的情况要比这复杂得多。

我们假设一下机器A上的管理员为了维护机器C，使用了一个FTP命令向

机器C进行远程登陆，那么在这个用HUB连接的网络里数据走向过程是这样的。首先机器A上的管理员输入的登陆机器C的FTP口令经过应用层FTP协议、传输层TCP协议、网络层IP协议、数据链路层上的以太网驱动程序一层一层的包裹，最后送到了物理层，我们的网线上。接下来数据帧送到了 HUB 上，现在由 HUB 向每一个接点广播由机器A发出的数据帧，机器B接收到由HUB广播发出的数据帧，并检查在数据帧中的地址是否和自己的地址相匹配，发现不是发向自己的后把这数据帧丢弃，不予理睬。而机器C也接收到了数据帧，并在比较之后发现是发现自己的，接下来他就对这数据帧进行分析处理。

在上面这个简单的例子中，机器B上的管理员如果很好奇，他很想知道究竟登陆机器C上FTP口令是什么？那么他要做的很简单，仅仅需要把自己机器上的网卡置于混杂模式，并对接收到的数据帧进行分析，从而找到包含在数据帧中的口令信息。

四 做一个自己的sniff

在上一节里，我们已经知道了 SNIFF 的基本原理是怎么一回事，这一节我们来亲自动手做一个自己的sniff，毕竟，用程序代码来说话比什么都要来得真实，也容易加深理解。

回头想一想我们上面说的原理，我们要做的事情有几件：

1. 把网卡置于混杂模式。

2. 捕获数据包。

3.分析数据包。

注：下面的源代码取至Chad Renfro的<< Basic Packet-SnifferConstruction from the Ground Up>>一文中（在这里不是让大家学习编程，而是让大家明白这个原理)。

```
/***********************Tcp_sniff_2.c********************/
1.#include
2.#include
3.#include
4.#include
5.#include
```

```
6.#include
7.#include
8.#include
9.#include "headers.h"
#define INTERFACE "eth0"
/*Prototype area*/
10.int Open_Raw_Socket(void);
11.int Set_Promisc(char *interface, intsock);
12.int main() {
13.int sock, bytes_recieved, fromlen;
14.char buffer[65535];
15.struct sockaddr_in from;
16.struct ip *ip;
17.struct tcp *tcp;
18.sock = Open_Raw_Socket();
19. Set_Promisc(INTERFACE, sock);
20. while(1)
22. {
23. fromlen = sizeof from;
24. bytes_recieved = recvfrom(sock, buffer, sizeofbuffer, 0, (struct
sockaddr *)&from, &fromlen);
25. printf("/nBytes received :::%5d/n",bytes_recieved);
26. printf("Source address :::%s/n",inet_ntoa(from.sin_addr));
27. ip = (struct ip *)buffer;
/*See if this is a TCP packet*/
28. if(ip->ip_protocol == 6) {
29. printf("IP header length :::%d/n",ip->ip_length);
30. printf("Protocol :::%d/n",ip->ip_protocol);
31. tcp = (struct tcp *)(buffer +(4*ip->ip_length));
```

```
32. printf("Source port :::%d/n",ntohs(tcp->tcp_source_port));
33. printf("Dest port :::%d/n",ntohs(tcp->tcp_dest_port));
34. }
35. }
36.}
37.int Open_Raw_Socket() {
38. int sock;
39. if((sock = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0){
/*Then the socket was not created properly and must die*/
40. perror("The raw socket was not created");
41. exit(0);
42. };
43. return(sock);
44. }
45.int Set_Promisc(char *interface, int sock ) {
46. struct ifreq ifr;
47. strncpy(ifr.ifr_name, interface,strnlen(interface)+1);
48. if((ioctl(sock, SIOCGIFFLAGS, &ifr) == -1)) {
/*Could not retrieve flags for the interface*/
49. perror("Could not retrive flags for the interface");
50. exit(0);
51. }
52. printf("The interface is ::: %s/n", interface);
53. perror("Retrieved flags from interface successfully");
54. ifr.ifr_flags |= IFF_PROMISC;
55. if (ioctl (sock, SIOCSIFFLAGS, &ifr) == -1 ) {
/*Could not set the flags on the interface */
56. perror("Could not set the PROMISC flag:");
57. exit(0);
```

```
58. }
59. printf("Setting interface ::: %s ::: to promisc",interface);
60. return(0);
61. }
/*********************EOF*********************************/
```

上面这段程序中有很详细的注解，不过我想还是有必要说一说，首先第 10 行--intOpen_Raw_Socket(void); 是我们的自定义函数，具体内容如下：

```
37.int Open_Raw_Socket() {
38. int sock;
39. if((sock = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0){
/*Then the socket was not created properly and must die*/
40. perror("The raw socket was not created");
41. exit(0);
42. };
43. return(sock);
44. }
```

第 39 行 if((sock = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0) { 这里我们调用了 socket 函数，使创建了了一个原始套接口，使之收到 TCP/IP 信息包。

接下来第 11 行-int Set_Promisc(char *interface, intsock)，这也是我们的自定义函数，目的是把网卡置于混杂模式，具体内容如下：

```
45.int Set_Promisc(char *interface, int sock ) {
46. struct ifreq ifr;
47. strncpy(ifr.ifr_name, interface,strnlen(interface)+1);
48. if((ioctl(sock, SIOCGIFFLAGS, &ifr) == -1)) {
/*Could not retrieve flags for the interface*/
49. perror("Could not retrive flags for the interface");
50. exit(0);
51. }
```

```
52. printf("The interface is ::: %s/n", interface);
53. perror("Retrieved flags from interface successfully");
54. ifr.ifr_flags |= IFF_PROMISC;
55. if (ioctl (sock, SIOCSIFFLAGS, &ifr) == -1 ) {
/*Could not set the flags on the interface */
56. perror("Could not set the PROMISC flag:");
57. exit(0);
58. }
59. printf("Setting interface ::: %s ::: to promisc",interface);
60. return(0);
61. }
```

首先 struct ifreq ifr; 定一了一个 ifrreg 的结构 ifr，接下来 strncpy(ifr.ifr_name,interface,strnlen(interface)+1);，就是把我们网络设备的名字填充到 ifr 结构中，在这里#define INTERFACE "eth0" ，让我们再往下看,ioctl(sock, SIOCGIFFLAGS,&ifr),SIOCGIFFLAGS 请求表示需要获取接口标志，现在到了第 54 行，在我们成功的获取接口标志后把他设置成混杂模式，ifr.ifr_flags|= IFF_PROMISC;ioctl (sock, SIOCSIFFLAGS,&ifr)。OK，现在我们所说的第一步已经完成--------把网卡置于混杂模式。

现在进入第二步，捕获数据包。从第 20 行开始，我们进入了一个死循环，while(1)，在第 24 行，recvfrom(sock,buffer, sizeof buffer, 0, (struct sockaddr *)&from,&fromlen)，这个函数要做的就是接收数据，冰把接收到的数据放入 buffer 中。就是这么简单，已经完成了我们要捕获数据包的任务。

到了第三步，分析数据包。27 行，ip = (struct ip*)buffer，使我们在头文件中的 IP 结构对应于所接收到的数据，接下来判断在网络层中是否使用的是 TCP 协议，if(ip->ip_protocol== 6) ，如果答案是，tcp 信息包从整个 IP/TCP 包 buffer +(4*ip->ip_length) 地址处开始，所以 31 行 tcp = (struct tcp*)(buffer +(4*ip->ip_length))，然后对应结构把你所需要的信息输出。

```
/************************headers.h*************************/
/*structure of an ip header*/
```

```
struct ip {
unsigned int ip_length:4; /*little-endian*/
unsigned int ip_version:4;
unsigned char ip_tos;
unsigned short ip_total_length;
unsigned short ip_id;
unsigned short ip_flags;
unsigned char ip_ttl;
unsigned char ip_protocol;
unsigned short ip_cksum;
unsigned int ip_source; unsigned int ip_dest;
};
/* Structure of a TCP header */
struct tcp {
unsigned short tcp_source_port;
unsigned short tcp_dest_port;
unsigned int tcp_seqno;
unsigned int tcp_ackno;
unsigned int tcp_res1:4, /*little-endian*/
tcp_hlen:4,
tcp_fin:1,
tcp_syn:1,
tcp_rst:1,
tcp_psh:1,
tcp_ack:1,
tcp_urg:1,
tcp_res2:2;
unsigned short tcp_winsize;
unsigned short tcp_cksum;
```

```
unsigned short tcp_urgent;
};
/********************EOF**********************************/
```

从上面的分析我们可以清楚的认识到，认识一个 SNIFF 需要对 TCP/IP 协议有着详细的了解，否则你根本无法找到你需要的信息。有了上面的基础，你可以自己来做一个你需要的 SNIFF 了。

五 常用的 SNIFF

很少有原因会让你自己亲自动手来做一个自己的 SNIFF，除非你是想了解他的原理，或者是其他一些特别的原因，比如你要在某个特殊的环境拦截一些特殊的数据包。下面我们就来看看一些在网络上经常使用的 SNIFF。

（1）windows 环境下

windows 环境下当然是大名鼎鼎的 netxray 以及 sniffer pro 了，实际上很多人都是用他在 windows 环境下抓包来分析，不过我想很少有人笨到去在别人的机器上安装一个图形界面的 SNIFF，除非他和管理员很熟悉........netxray 的使用就不多说了，反正 windows 下的东西就是 click，click，click，非常的方便用户。

（2）UNUX 环境下

UNUX 环境下的 sniff 可以说是百花齐放，一抓就是一大把,如 sniffit，snoop,tcpdump,dsniff 等都是比较常见的，他们都有一个好处就是发布源代码，可以让你研究，当然也都是免费的:）

1. sniffit

sniffit 可以运行在 Solaris、SGI 和 Linux 等平台上，由 LawrenceBerkeley Laboratory 实验室开发的一个免费的网络监听软件。最近 Sniffit0.3.7 也推出了 NT 版本，并也支持 WINDOWS2000.

使用方法:

-v 显示版本信息

-a 以 ASCII 形式将监听的结果输出。

-A 在进行记录时，所有不可打印的字符都用代替

-b 等同于同时使用参数-t & -s。

-d 将监听所得内容以十六进制方式显示在当前终端

-p 记录连接到的包，0 为所有端口。缺省为 0。

-P protocol 选择要检查的协议，缺省为 TCP。可能的选择有 IP、TCP、ICMP、UDP 和他们的组合。

-s 指定 sniffer 检查从 发送的数据包。 -t 指定 sniffer 检查发送到的数据包。

-i 进入交互模式

-l 设定数据包大小，default 是 300 字节

注：参数可以用@来表示一个 IP 范围，比如 -t 192.168.@ -t 和-s 只适用于 TCP/UDP 数据包，对于 ICMP 和 IP 也进行解释。但如果只选择了-p 参数，则只用于 TCP 和 UDP 包。

举例说明:

#sniffit -a -p 21 -t xxx.xxx.xxx.xxx

监听流向机器 xxx.xxx.xxx.xxx 的 21 端口(FTP)的信息,并以 ASCII 显示

#sniffit -d -p 23 -b xxx.xxx.xxx.xxx

监听所有流出或流入机器 xxx.xxx.xxx.xxx 的 23 端口(telnet)的信息，并以 16 进制显示你可以在这里找到 sniffithttp://reptile.rug.ac.be/~coder/sniffit/sniffit.html

2. snoop

snoop 默认情况安装在 Solaris 下，是一个用于显示网络交通的程序，不过 SNIFF 是把双刃剑，既然管理员能用他来监视自己的网络，当然一个心怀恶意的入侵者也可以用他来 SNIFF 自己感兴趣的内容。值得一提的是，SNOOP 被发现存在一个缓冲区溢出漏洞，当以导致入侵者以运行 snoop(通常为 root)的身份远程进入系统。这是题外话，暂且就此打住。

使用方法：

[-a] # Listen to packets on audio

[-d device] # settable to le?, ie?, bf?, tr?

[-s snaplen] # Truncate packets

[-c count] # Quit after count packets

[-P] # Turn OFF promiscuous mode

[-D] # Report dropped packets

[-S] # Report packet size

[-i file] # Read previously captured packets

[-o file] # Capture packets in file

[-n file] # Load addr-to-name table from file

[-N] # Create addr-to-name table

[-t r|a|d] # Time: Relative, Absolute or Delta

[-v] # Verbose packet display

[-V] # Show all summary lines

[-p first[,last]] # Select packet(s) to display

[-x offset[,length]] # Hex dump from offset for length

[-C] # Print packet filter code

例如：

#snoop -o saved A B

监听机器 A 与 B 的谈话，并把内容存储于文件 saved 中

3. tcpdump

tcpdmp 也算是一个很有名气的网络监听软件，FREEBSD 还把他附带在了系统上，是一个被很多 UNIX 高手认为是一个专业的网络管理工具。

使用方法:

tcpdump 采用命令行方式，它的命令格式为：

tcpdump [-adeflnNOpqStvx][-c 数量][-F 文件名][-i 网络接口][-r 文件名][-s snaplen][-T 类型][-w 文件名][表达式]

1. tcpdump 的选项介绍

-a　　将网络地址和广播地址转变成名字；

-d　　将匹配信息包的代码以人们能够理解的汇编格式给出；

-dd　　将匹配信息包的代码以 c 语言程序段的格式给出；

-ddd　　将匹配信息包的代码以十进制的形式给出；

-e　　在输出行打印出数据链路层的头部信息；

-f　　　将外部的 Internet 地址以数字的形式打印出来；

-l　　　使标准输出变为缓冲行形式；

-n　　　不把网络地址转换成名字；

-t　　　在输出的每一行不打印时间戳；

-v　　　输出一个稍微详细的信息，例如在 ip 包中可以包括 ttl 和服务类型的信息；

-vv　　　输出详细的报文信息；

-c　　　在收到指定的包的数目后，tcpdump 就会停止；

-F　　　从指定的文件中读取表达式,忽略其它的表达式；

-i　　　指定监听的网络接口；

-r　　　从指定的文件中读取包(这些包一般通过-w 选项产生)；

-w　　　直接将包写入文件中，并不分析和打印出来；

-T　　　将监听到的包直接解释为指定的类型的报文，常见的类型有 rpc 和 snmp

2. tcpdump 的表达式介绍

表达式是一个正则表达式，tcpdump 利用它作为过滤报文的条件，如果一个报文满足表达式的条件，则这个报文将会被捕获。如果没有给出任何条件，则网络上所有的信息包将会被截获。

在表达式中一般如下几种类型的关键字，一种是关于类型的关键字，主要包括 host，net，port,例如 host 210.27.48.2，指明 210.27.48.2 是一台主机，net 202.0.0.0 指明 202.0.0.0 是一个网络地址，port 23 指明端口号是 23。如果没有指定类型，缺省的类型是 host.

第二种是确定传输方向的关键字，主要包括 src , dst ,dst or src, dstand src ,这些关键字指明了 传输的方向。举例说明，src 210.27.48.2,指明 ip 包中源地址是 210.27.48.2 , dst net 202.0.0.0 指明目的网络地址是 202.0.0.0 。如果没有指明方向关键字，则缺省是 src ordst 关键字。

第三种是协议的关键字，主要包括 fddi,ip,arp,rarp,tcp,udp 等类型。Fddi 指明是在 FDDI(分布式光纤数据接口网络)上的特定的网络协议，实际上它是"ether"的别名，fddi 和 ether 具有类似的源地址和目的地址，所以可以将 fddi

协议包当作 ether 的包进行处理和分析。其他的几个关键字就是指明了监听的包的协议内容。如果没有指定任何协议，则 tcpdump 将会监听所有协议的信息包。

除了这三种类型的关键字之外，其他重要的关键字如下：gateway,broadcast,less,greater,还有三种 逻辑运算，取非运算是 'not '' ! ',与运算是'and','&&';或运算 是'or' ,'||'。

举例使用:

#tcpdump host AAA.BBB.CCC.DDD

将监听 IP 地址为 AAA.BBB.CCC.DDD 的机器的通话

#tcpdump tcp port 23 host AAA.BBB.CCC.DDD

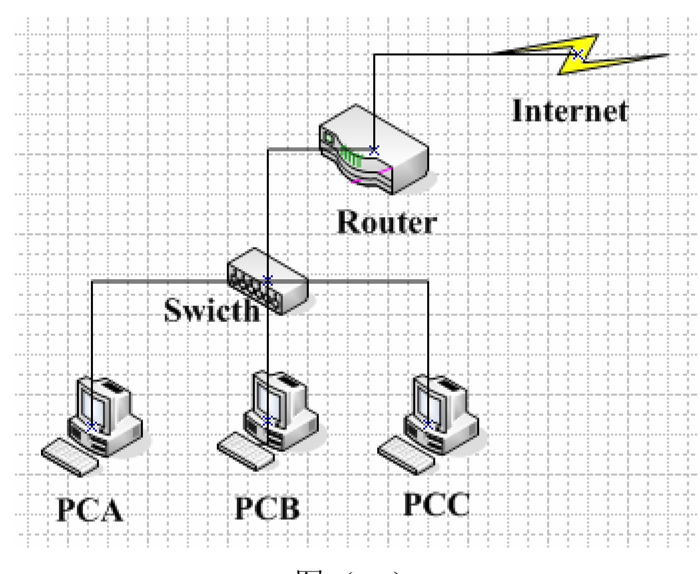
将监听 IP 地址为 AAA.BBB.CCC.DDD 的机器的 23 端口的通话

4. dsniff

之所以要谈谈 dsniff，是因为他不仅仅是一个 sniff，在他的整个套件包中，包含了很多其它有用的工具，如 arpspoof，dnsspoof，macof，tcpkill 等等，SNIFF 的手段更加的多样和复杂化。dsniff 是由 DugSong 开发的你可以在他的主页上找到这个工具。 目前 dsniff 支持 OpenBSD (i386),Redhat Linux (i386), 和 Solaris (sparc). 并且在 FreeBSD, DebianLinux, Slackware Linux, AIX,和 HP-UX 上也能运转得很好。但是 dsniff 需要几个其他的第三方软件进行支持，他们分别是，BerkeleyDB ，OpenSSL， libpcap， libnet，libnids。如果条件允许的话，你最好能够亲自读一读 dsniff 的源代码，你可以在 http://naughty.monkey.org/~dugsong/ 找到 dsniff。

六 深入 sniff

单纯的 sniff 的功能始终是局限的，所以在大多数的情况下，sniff 往往和其他手段结合起来使用，sniff 和 spoof 已及其他技术手段结合在一起对网络构成的危害是巨大的。单纯的 sniff 好比缺了一只腿，无法发挥大的作用，例如在 sniff 原理一节中我们讨论的例子里，我一再的强调我们使用的是一个普通的 HUB 进行连接是有原因的，如果我们把在图一中的 HUB 用一个 switch 代替，那情况就要复杂一些了，如图二所示:



图（二）

在图二中，我们的机器 A、B、C 与 Switch 相连接，而 Switch 通过路由器 Router 访问外部网络。我们先来了解 Switch 的工作原理：

在我们图一中的 HUB 只是简单地把所接收到的信号通过所有端口（除了信号来的那个口）重复发送出去不同，而图二中的 Switch 却可以检查每一个收到的数据包，并对数据包进行相应的处理。在 Switch 内保存着每一个网段上所有节点的物理地址，只允许必要的网络流量通过 Switch。举例来说，当 Switch 接收到一个数据包之后，根据自身保存的网络地址表检查数据包内包含的发送和接收方地址。如果接收方位于发送方网段，该数据包就会被 Switch 丢弃，不能通过交换机传送到其它的网段；如果接收方和发送方位于两个不同的网段，该数据包就会被 Switch 转发到目标网段。这样，通过交换机的过滤和转发，可以有效避免网络广播风暴，减少误包和错包的出现。顺便说一句，现在 Switch 和 HUB 的价格相去无几，所以 hub 正逐渐被网络交换机取代。

现在回到我们的例子中来，在图二中仍然和图一一样，我们假设机器 A 上的管理员为了维护机器 C，使用了一个 FTP 命令向机器 C 进行远程登陆，那么在这里，数据是这样走的：首先机器 A 上的管理员输入的登陆机器 C 的 FTP 口令经过应用层 FTP 协议、传输层 TCP 协议、网络层 IP 协议、数据链路层上的以太网驱动程序一层一层的包裹，最后送到了物理层，我们的网线上。接下来数据帧送到了 Switch 上，而 Switch 检查数据帧中的目的地址，并在他自身保存的网络地址表中知道了他应该把这数据帧发到机器 C 那里，于是，接下来机器 C 接收到了从 A 发来的信息，发现他是发给自己的信息，于是进行分析处理。

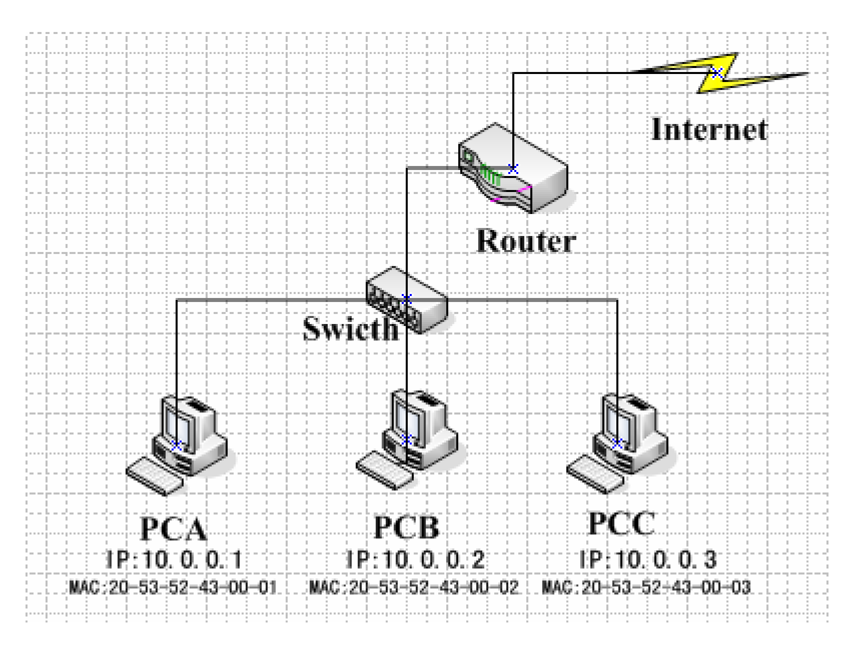
OK，现在我们机器B上的管理员的好奇心只能深深的埋藏在心里了，因为数据包根本就没有经过他，就算他把自己的网卡设置成混杂模式也是有力无处使。

在了解在一个Switch环境下原理后，我们结合一些手段去设法sniff，是的，我们可以做到这一点，有许多的手段可以让管理员B满足他的好奇心，在下面我会提出几个办法，当然只是其中的一些办法而已。

1 ARP Spoof

在基于IP通信的内部网中，我们可以使用 ARP Spoof 的手段，了解什么是ARPSpoof的前提你先要明白一下什么是ARP和RARP协议，什么是MAC地址，什么又是 IP 地址。ARP 协议是地址转换协议，RARP 被称为反向地址转换协议，他们负责把IP地址和MAC地址进行相互转换对应。

ARP Spoof 攻击的根本原理是因为计算机中维护着一个 ARP 高速缓存，并且这个ARP高速缓存是随着计算机不断的发出ARP请求和收到ARP响应而不断的更新的，ARP 高速缓存的目的是把机器的 IP 地址和 MAC 地址相互映射。你可以使用 arp 命令来查看你自己的 ARP 高速缓存。现在设想一下，一个Switch工作在数据链路层，他根据MAC地址来转发他所接收的数据包，而计算器维护的ARP高速缓存却是动态的......你想到什么了吗？



为了加深理解，现在给我们的机器编上号（如上图），机器 A：IP 地址为10.0.0.1,MAC 地址为 20-53-52-43-00-01 ,机器 B：IP 地址为 10.0.0.2 ,MAC

地址为 20-53-52-43-00-02,机器 C：IP 地址为 10.0.0.3 ,MAC 地址为 20-53-52-43-00-03 。现在机器 B 上的管理员是个聪明的家伙，他向机器 A 发出一个 ARP Reply（协议没有规定一定要等 ARP Request 出现才 能发送 ARPReply，也没有规定一定要发送过 ARP Request 才能接收 ARPReply），其中的目的 IP 地址为 10.0.0.1,目的 MAC 地址为 20-53-52-43-00-01 ,而源 IP 地址为 10.0.0.3，源 MAC 地址为 20-53-52-43-00-02 ，好了，现在机器 A 更新了他的 ARP 高速缓存，并相信了 IP 地址为 10.0.0.3 的机器的 MAC 地址是 20-53-52-43-00-02 。当机器 A 上的管理员发出一条 FTP 命令时---ftp 10.0.0.3，数据包被送到了 Switch，Switch 查看数据包中的目的地址，发现 MAC 为 20-53-52-43-00-02，于是，他把数据包发到了机器 B 上。再设想一下，如果不想影响 A 和 C 之间的通信该怎么办？你可以同时欺骗他们双方，来一个 man-in-middle 。

当然，在实际的操作中你还需要考虑到一些其他的事，比如某些操作系统在会主动的发送 ARP 请求包来更新相应的 ARP 入口等。

2. MAC Flooding

在上面我们曾经提到过，Switch 之所以能够由数据包中目的 MAC 地址判断出他应该把数据包发送到那一个端口上是根据他本身维护的一张地址表。这张地址表可能是动态的也可能是静态的，这要看 Switch 的厂商和 Switch 的型号来定，对于某些 Switch 来说，他维护的是一张动态的地址表，并且地址表的大小是有上限的，比如 3comSuperstack Switch 3300 （3c16981 Hardware v.1 Softwarev.2.10） 就是这样一种 Switch，我们可以通过发送大量错误的地址信息而使 SWITCH 维护的地址表“溢出”，从而使他变成广播模式来达到我们要 sniff 机器 A 与机器 C 之间的通信的目的。

3. Fake the MAC address

伪造 MAC 地址也是一个常用的办法，不过这要基于你网络内的 Switch 是动态更新其地址表，这实际上和我们上面说到的 ARP Spoof 有些类似，只不过现在你是想要 Switch 相信你，而不是要机器 A 相信你。因为 Switch 是动态更新其地址表的，你要做的事情就是告诉 Switch：HI,我是机器 C。换成技术上的问题你只不过需要向 Switch 发送伪造过的数据包，其中源 MAC 地址对应的是机器 C 的 MAC 地址，现在 Switch 就把机器 C 和你的端口对应起来了。不过其中存在一

个问题，现在机器 C 也会说了：HI，Switch 老大，我才是机器 C 呢!，现在你该怎么办？切，还用问！让他说不了话就可以了，DOS 还是其他什么，随便你了......

4. ICMP Router Advertisements

这主要是由 ICMP 路由器发现协议(IRDP)的缺陷引起的，在 Windows95、98、2000 及 SunOS、Solaris2.6 等系统中，都使用了 IRDP 协议，SunOS 系统只在某些特定的情况下使用该协议，而 Windows95,Windows95b, Windows98, Windows98se, 和 Windows2000 都是默认的使用 IRDP 协议。IRDP 协议的主要内容就是告诉人们谁是路由器，设想一下，一个 HACK 利用 IRDP 宣称自己是路由器的情况会有多么的糟糕！所有相信 HACK 的请求的机器把他们所有的数据都发送给 HACK 所控制的机器.........

5. ICMP Redirect

所谓 ICMP 重定向，就是指告诉机器向另一个不同的路由发送他的数据包，ICMP 重定向通常使用在这样的场合下，假设 A 与 B 两台机器分别位于同一个物理网段内的两个逻辑子网内，而 A 和 B 都不知道这一点，只有路由器知道，当 A 发送给 B 的数据到达路由器的时候，路由器会向 A 送一个 ICMP 重定向包裹，告诉 A：HI，别再送数据给我转交了，你就直接送到 B 那里就可以了。设想一下，一个 hack 完全可以利用这一点，使得 A 发送给 B 的数据经过他。

上面提到的这些方法只不是其中的一些，为了配合 sniff 能够工作得更有效率，还有其他许多的办法，其实 sniff 的目的说穿了只有一个，就是抓包，从抓包这个概念上引伸下去，所有为了能够抓到网络上的信息包而采用的技术都可以归入 sniff，单纯的 sniff 是没有什么效率的。你还能想到什么吗？进攻路由器，在路由器上放置 sniff......，在系统内核中植入 sniff......等等。

七 如何防止 SNIFF

防止 sniff 最有效的手段就是进行合理的网络分段，并在网络中使用交换机和网桥，在理想的情况下使每一台机器都拥有自己的网络段，当然这会使你的网络建设费用增加很多，所以你可以尽量使相互信任的机器属于同一个网段，使他们互相之间不必担心 sniff 的存在。并在网段于网段间进行硬件屏障。你也可以使用加密技术对你在网络中传送的敏感数据如户 ID 或口令，你的银行帐号，

商业机密等进行加密，你可以选用ＳＳＨ等加密手段。为了防止 ARP 欺骗，你可以使用永久的ＡＲＰ缓存条目，反正上面的攻击手段和原理你也看了，你就反过来想想该怎么办好了。不过有盾必有矛，平时的安全意识才是最重要的。

（注：以下关于 AntiSniff 的介绍取至 backend 翻译整理的 L0pht AntiSniff 技术文档一文）

当你做做层层保护后，你还是怀疑自己的网络上存在 sniff 该怎么办？L0pht 小组为了探测 sniff 专门发布了一个软件 AntiSniff，当然这个软件不是免费的:）,AntiSniff 工具用于检测局域网中是否有机器处于混杂模式，AntiSniff Version1.x 被设计为运行在以太网的 Windows 系统中，提供了简单易用的图形用户界面，AntiSniffVersion1.x 主要工作在非交换环境下的本地网段中，如果运行在交换环境下其功能将大打折扣。AntiSniffVer 2.0 将不但能在本地网段中，而且能够穿过路由器和交换机进行工作。

◆ 操作系统类特殊测试

Linux 内核测试

旧版本的 Linux 内核存在一个奇怪的特性，可被用于确定机器是否处于混杂模式。在正常情形下，网卡会过滤和丢弃那些目标地址不是本机 MAC 地址或以太网广播地址的数据包。如果数据包的目标地址为本机以太网地址或广播地址，将传送给内核进行处理，因为其认为该以太网数据帧包含了本机的正确 IP 地址或该网络广播地址。如果网卡处于混杂模式，则每个数据包都会传递给操作系统进行分析或处理。许多版本的 Linux 内核只检查数据包中的 IP 地址以确定是否存放到 IP 堆栈中进行处理。为了利用这一点，AntiSniff 构造一个无效以太网地址而 IP 地址有效的数据包。对于使用了这些内核版本和处于混杂模式的 Linux 系统，由于只检查到 IP 地址有效而将其接收并存放到相应堆栈中。通过在这个伪造的以太网数据帧中构造一个 ICMPECHO 请求，这些系统会返回响应包(如果处于混杂模式)或忽略(如果不处于混杂模式)，从而暴露其工作模式。当伪造的以太网数据帧中的 IP 地址设置为网络广播地址时这个测试非常有效。AntiSniff 的使用者可以修改伪造的以太网址，缺省值为 66:66:66:66:66:66。

NetBSD

许多 NetBSD 内核具有与上述 Linux 内核相同的特性，不过伪造以太网数据

帧中的 IP 地址必须设为广播地址。

Windows 95/98/NT

根据对网络驱动程序头文件的了解，可以知道当处于混杂模式时，Microsoft 的操作系统会确切地检查每个包的以太网地址。如果与网卡的以太网地址匹配，将作为目标 IP 地址为本机的数据包存放到相应堆栈中处理。可以被利用的一点是系统对以太网广播包的分析。在正常情形下，例如机器工作在非混杂模式下，网卡只向系统内核传输那些目标以太网址与其匹配或为以太网广播地址(ff:ff:ff:ff:ff:ff)的数据包。如果机器处于混杂模式下，网络驱动程序仍然会检查每个数据包的以太网地址，但检查是否为广播包时却只检查头 8 位地址是否为 0xff。因此，为了使处于混杂模式的系统返回响应信息，AntiSniff 构造以太网地址为 ff:00:00:00:00:00 且含有正确目标 IP 地址的数据包，当 Microsoft 的操作系统接收到这个数据包时，将根据网络驱动程序检查到的细微差别而返回响应包(如果处于混杂模式)或丢弃这个数据包(如果处于非混杂模式)。

需要注意的是，这个检查与使用的网络驱动程序有关。Microsoft 缺省的网络驱动程序具有以上特性，大多数的厂商为了保持兼容性也继承了这些特性。不过有些网卡会在其硬件层中检查以太网地址的头 8 位，所以可能会无论系统真正的状态是什么都总是返回正值。关于这类网卡和驱动程序请访问 AntiSniff Ver1.x 的 web 网站。

◆ DNS 测试

进行 DNS 测试的原因是许多攻击者使用的网络数据收集工具都对 IP 地址进行反向 DNS 解析，因为他们希望根据域名寻找更有价值的主机。例如 joepc1.foo.bar 对攻击者的吸引力往往不如 payroll.foo.bar 这种商业域名。此时这些工具就由被动型网络工具变为主动型网络工具了。而不监听网络通讯的机器不会试图反向解析数据包中的 IP 地址。为了利用这一点，AntiSniff Ver1.x 使自身处于混杂模式下，向网络发送虚假目标 IP 地址的数据包，然后监听是否有机器发送该虚假目标 IP 地址的反向 DNS 查询。伪造数据包的以太网地址、检查目标、虚假目标 IP 地址可由用户定制。

◆ 网络和主机响应时间测试

这种测试已被证明是最有效的。它能够发现网络中处于混杂模式的机器，而不管其操作系统是什么。警告，这个测试会在很短的时间内产生巨大的网络通讯流量。进行这种测试的理由是不处于混杂模式的网卡提供了一定的硬件底层过滤机制。也就是说，目标地址非本地(广播地址除外)的数据包将被网卡的固件丢弃。在这种情况下，骤然增加、但目标地址不是本地的网络通讯流量对操作系统的影响只会很小。而处于混杂模式下的机器则缺乏此类底层的过滤，骤然增加、但目标地址不是本地的网络通讯流量会对该机器造成较明显的影响(不同的操作系统/内核/用户方式会有不同)。这些变化可以通过网络通讯流量工具监视到。

根据以上要点，AntiSniff Ver 1.x 首先利用 ICMPECHO 请求及响应计算出需要检测机器的响应时间基准和平均值。在得到这个数据后，立刻向本地网络发送大量的伪造数据包。与此同时再次发送测试数据包以确定平均响应时间的变化值。非混杂模式的机器的响应时间变化量会很小，而混杂模式的机器的响应时间变化量则通常会有 1-4 个数量级。为了对付攻击者和入侵者们最常用的多种工具，AntiSniff 进行了三种网络饱和度测试：SIXTYSIX、TCPSYN 和 THREEWAY。

*SIXTYSIX 测试构造的数据包数据全为 0x66。这些数据包不会被非混杂模式的机器接收，同时方便使用常见的网络监听/分析工具(如 tcpdump 和 snoop 等)记录和捕获。

*TCPSYN 测试构造的数据包包含有效的 TCP 头和 IP 头，同时 TCP 标志域的 SYN 位被设置。

*THREEWAY 测试采取的原理基本上与 TCPSYN 一样，但更复杂些。在这种测试中两个实际不存在的机器间多次建立完整的 TCP 三方握手通讯。它能够更好地欺骗那些骇客工具。

AntiSniff Ver 1.x 中能够通过以上三种数据包测试发现正处于混杂模式机器的测试方法最好周期性地进行和与以前的数据比较。响应时间测试第一次运行的数据还能够用于分析一个大型网络在 flooding 和非 flooding 状态时的性能，并帮助工程师调整网络性能。一旦确信本地网络已运行在正常(没有未经允许而处于混杂模式的机器)状态，就应该设置 AntiSniff 工具周期性运行。只要发现某台机器性能(响应时间)发生数量级的变化，一般就能确定其正处于混杂模式。这种方法不需比较两台独立系统间的性能数据，而只需比较同一台机器不同时候

的数据就能确定该机器是否处于混杂模式。

八　结尾

本文旨在向你描述 sniff 的基本原理，为的是要使你不仅仅能够了解什么是 sniff 而已，而是要明白 sniff 运转的根本原理，文章参考了大量的资料，牵涉到直接引用的已经注明出处和作者，非常的感谢他们。在此还要感谢 W.Richhard.Stevens,虽然其人已逝，但留下的 TCP/IP 三卷本真是造福了大家，文章的很多地方也是拜他老人家指点迷经才得以感悟。最后还要感谢雀巢咖啡，让我得以熬夜把这篇文章写完，呵呵，谢谢大家。

说了这么多不知大家明白了没有，嗅探的基本原理，其中还涉及到了反嗅探的一些知识，虽说不说但足以让大家够用，好了不多说了，休息一下，放松一下眼疲劳，顺便再把刚才的知识回忆一下，呆会儿我们继续下一章节的内容。

注意：嗅探的基本原理是你明白 sniffer 工作的重要方法，因此可能会重复出现，在些向大家申明，如果大家明白了，大可以跳过这些章节以便在后面可以更快的进行学习！

SNIFFER（嗅探器）-简介

SNIFFER（嗅探器）-简介

Sniffer（嗅探器）是一种常用的收集有用数据方法，这些数据可以是用户的帐号和密码，可以是一些商用机密数据等等。Snifffer 可以作为能够捕获网络报文的设备,ISS 为 Sniffer 这样定义:Sniffer 是利用计算机的网络接口截获目的地为其他计算机的数据报文的一种工具。

Sniffer 的正当用处主要是分析网络的流量,以便找出所关心的网络中潜在的问题。例如,假设网络的某一段运行得不是很好,报文的发送比较慢,而我们又不知道问题出在什么地方,此时就可以用嗅探器来作出精确的问题判断。在合理的网络中,sniffer 的存在对系统管理员是致关重要的,系统管理员通过 sniffer 可以诊断出大量的不可见模糊问题,这些问题涉及两台乃至多台计算机之间的异常通讯有些甚至牵涉到各种的协议,借助于 sniffer 系统管理员可以方便的确定出多少的通讯量属于哪个网络协议、占主要通讯协议的主机是哪一台、大多数通讯目的地是哪台主机、报文发送占用多少时间、或着相互主机的报文传送间隔时间等等,这些信息为管理员判断网络问题、管理网络区域提供了非常宝贵的信息。

嗅探器与一般的键盘捕获程序不同。键盘捕获程序捕获在终端上输入的键值，而嗅探器则捕获真实的网络报文。

为了对 sniffer 的工作原理有一个深入的了解，我们先简单介绍一下 HUB 与网卡的原理。

预备知识

HUB 工作原理

由于以太网等很多网络（常见共享 HUB 连接的内部网）是基于总线方式，物理上是广播的，就是当一个机器发给另一个机器的数据，共享 HUB 先收到然后把它接收到的数据再发给其他的（来的那个口不发了）每一个口，所以在共享 HUB 下面同一网段的所有机器的网卡都能接收到数据。

交换式 HUB 的内部单片程序能记住每个口的 MAC 地址,以后就该哪个机器接收就发往哪个口，而不是像共享 HUB 那样发给所有的口，所以交换 HUB 下只有该接收数据的机器的网卡能接收到数据，当然广播包还是发往所有口。显然共享 HUB 的工作模式使得两个机器传输数据的时候其他机器别的口也占用了，所以共

享HUB决定了同一网段同一时间只能有两个机器进行数据通信，而交换HUB两个机器传输数据的时候别的口没有占用，所以别的口之间也可以同时传输。这就是共享HUB与交换HUB不同的两个地方，共享HUB是同一时间只能一个机器发数据并且所有机器都可以接收，只要不是广播数据交换HUB同一时间可以有对机器进行数据传输并且数据是私有的。

网卡工作原理

再讲讲网卡的工作原理。网卡收到传输来的数据，网卡内的单片程序先接收数据头的目的MAC地址，根据计算机上的网卡驱动程序设置的接收模式判断该不该接收，认为该接收就在接收后产生中断信号通知CPU，认为不该接收就丢弃不管，所以不该接收的数据网卡就截断了，计算机根本就不知道。CPU得到中断信号产生中断，操作系统就根据网卡驱动程序中设置的网卡中断程序地址调用驱动程序接收数据，驱动程序接收数据后放入信号堆栈让操作系统处理。

局域网如何工作

数据在网络上是以很小的称为帧(Frame)的单位传输的帧由好几部分组成，不同的部分执行不同的功能。(例如，以太网的前12个字节存放的是源和目的的地址，这些位告诉网络：数据的来源和去处。以太网帧的其他部分存放实际的用户数据、TCP/IP的报文头或IPX报文头等等)。

帧通过特定的网络驱动程序进行成型，然后通过网卡发送到网线上。通过网线到达它们的目的机器，在目的机器的一端执行相反的过程。接收端机器的以太网卡捕获到这些帧，并告诉操作系统帧的到达，然后对其进行存储。就是在这个传输和接收的过程中，嗅探器会造成安全方面的问题。

通常在局域网（LAN）中同一个网段的所有网络接口都有访问在物理媒体上传输的所有数据的能力，而每个网络接口都还应该有一个硬件地址，该硬件地址不同于网络中存在的其他网络接口的硬件地址，同时，每个网络至少还要一个广播地址。(代表所有的接口地址)，在正常情况下，一个合法的网络接口应该只响应这样的两种数据帧：

1、帧的目标区域具有和本地网络接口相匹配的硬件地址。

2、帧的目标区域具有“广播地址”。

在接受到上面两种情况的数据包时，网卡通过cpu产生一个硬件中断，

该中断能引起操作系统注意，然后将帧中所包含的数据传送给系统进一步处理。

当采用共享 HUB，用户发送一个报文时，这些报文就会发送到 LAN 上所有可用的机器。在一般情况下，网络上所有的机器都可以“听”到通过的流量，但对不属于自己的报文则不予响应（换句话说，工作站 A 不会捕获属于工作站 B 的数据，而是简单的忽略这些数据）。

如果局域网中某台机器的网络接口处于杂收（promiscuous）模式（即网卡可以接收其收到的所有数据包，下面会详细地讲），那么它就可以捕获网络上所有的报文和帧，如果一台机器被配置成这样的方式，它（包括其软件）就是一个嗅探器。

Sniffer

Sniffer 原理

有了这 HUB、网卡的工作原理就可以开始讲讲 SNIFFER。首先，要知道 SNIFFER 要捕获的东西必须是要物理信号能收到的报文信息。显然只要通知网卡接收其收到的所有包（一般叫作杂收 promiscuous 模式：指网络上的所有设备都对总线上传送的数据进行侦听，并不仅仅是它们自己的数据。），在共享 HUB 下就能接收到这个网段的所有包，但是交换 HUB 下就只能是自己的包加上广播包。

要想在交换 HUB 下接收别人的包，那就要让其发往你的机器所在口。交换 HUB 记住一个口的 MAC 是通过接收来自这个口的数据后并记住其源 MAC，就像一个机器的 IP 与 MAC 对应的 ARP 列表，交换 HUB 维护一个物理口（就是 HUB 上的网线插口，这之后提到的所有 HUB 口都是指网线插口）与 MAC 的表，所以可以欺骗交换 HUB 的。可以发一个包设置源 MAC 是你想接收的机器的 MAC，那么交换 HUB 就把你机器的网线插的物理口与那个 MAC 对应起来了，以后发给那个 MAC 的包就发往你的网线插口了，也就是你的网卡可以 SNIFFER 到了。注意这物理口与 MAC 的表与机器的 ARP 表一样是动态刷新的，那机器发包后交换 HUB 就又记住他的口了，所以实际上是两个在争，这只能应用在只要收听少量包就可以的场合。

内部网基于 IP 的通信可以用 ARP 欺骗别人机器让其发送给你的机器，如果要想不影响原来两方的通信，可以欺骗两方，让其都发给你的机器再由你的机器转发，相当于做中间人，这用 ARP 加上编程很容易实现。并且现在很多设备支持远程管理，有很多交换 HUB 可以设置一个口监听别的口，不过这就要管理权限了。

利用这一点，可以将一台计算机的网络连接设置为接受所有以太网总线上的数据，从而实现 sniffer。Sniffer 就是一种能将本地网卡状态设成‘杂收’状态的软件，当网卡处于这种“杂收”方式时，该网卡具备“广播地址”，它对遇到的每一个帧都产生一个硬件中断以便提醒操作系统处理流经该物理媒体上的每一个报文包。（绝大多数的网卡具备置成杂收方式的能力）

可见，sniffer 工作在网络环境中的底层，它会拦截所有的正在网络上传送的数据，并且通过相应的软件处理，可以实时分析这些数据的内容，进而分析所处的网络状态和整体布局。值得注意的是：sniffer 是极其安静的，它是一种消极的安全攻击。

嗅探器在功能和设计方面有很多不同。有些只能分析一种协议，而另一些可能能够分析几百种协议。一般情况下，大多数的嗅探器至少能够分析下面的协议：标准以太网、TCP/IP、IPX、DECNet。

嗅探器造成的危害

Sniffing 是作用在网络基础结构的底层。通常情况下， 用户并不直接和该层打交道，有些甚至不知道有这一层存在。所以，应该说 snffer 的危害是相当之大的，通常，使用 sniffer 是在网络中进行欺骗的开始。它可能造成的危害：

嗅探器能够捕获口令。这大概是绝大多数非法使用 sniffer 的理由，sniffer 可以记录到明文传送的 userid 和 passwd。

能够捕获专用的或者机密的信息。比如金融帐号，许多用户很放心在网上使用自己的信用卡或现金帐号，然而 sniffer 可以很轻松截获在网上传送的用户姓名、口令、信用卡号码、截止日期、帐号和 pin。比如偷窥机密或敏感的信息数据，通过拦截数据包，入侵者可以很方便记录别人之间敏感的信息传送，或者干脆拦截整个的 email 会话过程。

可以用来危害网络邻居的安全，或者用来获取更高级别的访问权限。

窥探低级的协议信息。

这是很可怕的事，通过对底层的信息协议记录，比如记录两台主机之间的网络接口地址、远程网络接口 ip 地址、ip 路由信息和 tcp 连接的字节顺序号码等。这些信息由非法入侵的人掌握后将对网络安全构成极大的危害，通常有人用 sniffer 收集这些信息只有一个原因：他正要进行一次欺骗（通常的 ip 地址欺

骗就要求你准确插入 tcp 连接的字节顺序号)，如果某人很关心这个问题，那么 sniffer 对他来说只是前奏，今后的问题要大得多。(对于高级的 hacker 而言，我想这是使用 sniffer 的唯一理由吧)

事实上，如果你在网络上存在非授权的嗅探器就意味着你的系统已经暴露在别人面前了。

一般 Sniffer 只嗅探每个报文的前 200 到 300 个字节。用户名和口令都包含在这一部分中，这是我们关心的真正部分。工人，也可以嗅探给定接口上的所有报文，如果有足够的空间进行存储，有足够的那里进行处理的话，将会发现另一些非常有趣的东西……

简单的放置一个嗅探器并将其放到随便什么地方将不会起到什么作用。将嗅探器放置于被攻击机器或网络附近，这样将捕获到很多口令，还有一个比较好的方法就是放在网关上。sniffer 通常运行在路由器，或有路由器功能的主机上。这样就能对大量的数据进行监控。sniffer 属第二层次的攻击。通常是攻击者已经进入了目标系统，然后使用 sniffer 这种攻击手段，以便得到更多的信息。

sniffer 除了能得到口令或用户名外，还能得到更多的其他信息，比如一个其他重要的信息，在网上传送的金融信息等等。sniffer 几乎能得到任何以太网上的传送的数据包。黑客会使用各种方法，获得系统的控制权并留下再次侵入的后门，以保证 sniffer 能够执行。在 Solaris 2.x 平台上，sniffer 程序通常被安装在/usr/bin 或/dev 目录下。黑客还会巧妙的修改时间，使得 sniffer 程序看上去是和其它系统程序同时安装的。

大多数以太网 sniffer 程序在后台运行，将结果输出到某个记录文件中。黑客常常会修改 ps 程序，使得系统管理员很难发现运行的 sniffer 程序。

以太网 sniffer 程序将系统的网络接口设定为混合模式。这样，它就可以监听到所有流经同一以太网网段的数据包，不管它的接受者或发送者是不是运行 sniffer 的主机。 程序将用户名、密码和其它黑客感兴趣的数据存入 log 文件。黑客会等待一段时间 ----- 比如一周后，再回到这里下载记录文件。

讲了这么多，那么到底我们可以用什么通俗的话来介绍 sniffer 呢？

计算机网络与电话电路不同，计算机网络是共享通讯通道的。共享意味着计算机能够接收到发送给其它计算机的信息。捕获在网络中传输的数据信息就称为

sniffing（窃听）。

以太网是现在应用最广泛的计算机连网方式。以太网协议是在同一回路向所有主机发送数据包信息。数据包头包含有目标主机的正确地址。一般情况下只有具有该地址的主机会接受这个数据包。如果一台主机能够接收所有数据包，而不理会数据包头内容，这种方式通常称为"混杂" 模式。

由于在一个普通的网络环境中，帐号和口令信息以明文方式在以太网中传输， 一旦入侵者获得其中一台主机的 root 权限，并将其置于混杂模式以窃听网络数据，从而有可能入侵网络中的所有计算机。

一句话，sniffer 就是一个用来窃听的黑客手段和工具。

再次叙述一遍，通常在同一个网段的所有网络接口都有访问在物理媒体上传输的所有数据的能力，而每个网络接口都还应该有一个硬件地址，该硬件地址不同于网络中存在的其他网络接口的硬件地址，同时，每个网络至少还要一个广播地址。（代表所有的接口地址），在正常情况下，一个合法的网络接口应该只响应这样的两种数据帧：

1、帧的目标区域具有和本地网络接口相匹配的硬件地址。

2、帧的目标区域具有"广播地址"。

在接受到上面两种情况的数据包时，nc 通过 cpu 产生一个硬件中断，该中断能引起操作系统注意，然后将帧中所包含的数据传送给系统进一步处理。

而 sniffer 就是一种能将本地 nc 状态设成（promiscuous）状态的软件，当 nc 处于这种"混杂"方式时，该 nc 具备"广播地址"，它对所有遭遇到的每一个帧都产生一个硬件中断以便提醒操作系统处理流经该物理媒体上的每一个报文包。（绝大多数的 nc 具备置成 promiscuous 方式的能力）

可见，sniffer 工作在网络环境中的底层，它会拦截所有的正在网络上传送的数据，并且通过相应的软件处理，可以实时分析这些数据的内容，进而分析所处的网络状态和整体布局。值得注意的是：sniffer 是极其安静的，它是一种消极的安全攻击。

通常 sniffer 所要关心的内容可以分成这样几类：

1、口令

我想这是绝大多数非法使用 sniffer 的理由，sniffer 可以记录到明文传送的

userid 和 passwd.就算你在网络传送过程中使用了加密的数据，sniffer 记录的数据一样有可能使入侵者在家里边吃肉串边想办法算出你的算法。

2、金融帐号

许多用户很放心在网上使用自己的信用卡或现金帐号，然而 sniffer 可以很轻松截获在网上传送的用户姓名、口令、信用卡号码、截止日期、帐号和 pin.

3、偷窥机密或敏感的信息数据

通过拦截数据包，入侵者可以很方便记录别人之间敏感的信息传送，或者干脆拦截整个的 email 会话过程。

3、窥探低级的协议信息。

这是很可怕的事，我认为，通过对底层的信息协议记录，比如记录两台主机之间的网络接口地址、远程网络接口 ip 地址、ip 路由信息和 tcp 连接的字节顺序号码等。这些信息由非法入侵的人掌握后将对网络安全构成极大的危害，通常有人用 sniffer 收集这些信息只有一个原因：他正在进行一次欺诈，（通常的 ip 地址欺诈就要求你准确插入 tcp 连接的字节顺序号,这将在以后整理的文章中指出）如果某人很关心这个问题，那么 sniffer 对他来说只是前奏，今后的问题要大得多。（对于高级的 hacker 而言，我想这是使用 sniffer 的唯一理由吧）

Sniffer 的工作环境

snifffer 就是能够捕获网络报文的设备。嗅探器的正当用处在于分析网络的流量,以便找出所关心的网络中潜在的问题。例如,假设网络的某一段运行得不是很好,报文的发送比较慢,而我们又不知道问题出在什么地方,此时就可以用嗅探器来作出精确的问题判断。

嗅探器在功能和设计方面有很多不同。有些只能分析一种协议，而另一些可能能够分析几百种协议。一般情况下,大多数的嗅探器至少能够分析下面的协议：

1.标准以太网

2.TCP/IP

3.IPX

4.DECNet

嗅探器通常是软硬件的结合。专用的嗅探器价格非常昂贵。另一方面，免费的嗅探器虽然不需要花什么钱，但得不到什么支持。

嗅探器与一般的键盘捕获程序不同。键盘捕获程序捕获在终端上输入的键值，而嗅探器则捕获真实的网络报文。嗅探器通过将其置身于网络接口来达到这个目的——例如将以太网卡设置成杂收模式。（为了理解杂收模式是怎么回事，先解释局域网是怎么工作的）。

数据在网络上是以很小的称为帧(Ftame)的单位传输的帧由好几部分组成，不同的部分执行不同的功能。（例如，以太网的前 12 个字节存放的是源和目的的地址，这些位告诉网络：数据的来源和去处。以太网帧的其他部分存放实际的用户数据、TCP/IP 的报文头或 IPX 报文头等等）。

帧通过特定的称为网络驱动程序的软件进行成型，然后通过网卡发送到网线上。通过网线到达它们的目的机器，在目的机器的一端执行相反的过程。接收端机器的以太网卡捕获到这些帧，并告诉操作系统帧的到达，然后对其进行存储。就是在这个传输和接收的过程中，嗅探器会造成安全方面的问题。

每一个在 LAN 上的工作站都有其硬件地址。这些地址唯一地表示着网络上的机器（这一点于 Internet 地址系统比较相似）。当用户发送一个报文时，这些报文就会发送到 LAN 上所有可用的机器。

在一般情况下，网络上所有的机器都可以“听”到通过的流量，但对不属于自己的报文则不予响应（换句话说，工作站 A 不会捕获属于工作站 B 的数据，而是简单的忽略这些数据）。

如果某在工作站的网络接口处于杂收模式，那么它就可以捕获网络上所有的报文和帧，如果一个工作站被配置成这样的方式，它（包括其软件）就是一个嗅探器。

嗅探器可能造成的危害：

1.嗅探器能够捕获口令

2.能够捕获专用的或者机密的信息

3.可以用来危害网络邻居的安全，或者用来获取更高级别的访问权限

事实上，如果你在网络上存在非授权的嗅探器就以为着你的系统已经暴露在别人面前了。（大家可以试试天行 2 的嗅探功能）

一般我们只嗅探每个报文的前 200 到 300 个字节。用户名和口令都包含在这一部分中，这是我们关心的真正部分。工人，也可以嗅探给定接口上的所有报文，

如果有足够的空间进行存储，有足够的那里进行处理的话，将会发现另一些非常有趣的东西……

简单的放置一个嗅探器宾将其放到随便什么地方将不会起到什么作用。

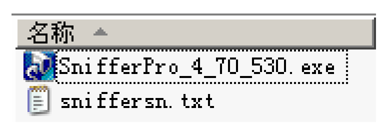
Sniffer 使用简介(一)

在这一章节里我们要学习到的是从 sniffer 的安装到 sniffer 的基本使用，这是对 sniffer 熟悉的一个过程，因此在这里请大家不要忘记，你首先要估的准备工作就是自已要拥有 sniffer 安装程序与安装序列号，当然如果你不喜欢英文界面你还可以拥有一个汉化包可以使你的 sniffer 有中文版式，那么你还等什么，去下载或者去购买都可以，如果你还不会，那么建议你用 baidu 或是 google 找找吧，我不好把地址写出来，现在互联网更新太快了，变化也太快了，所以请大家自已动手去找找。

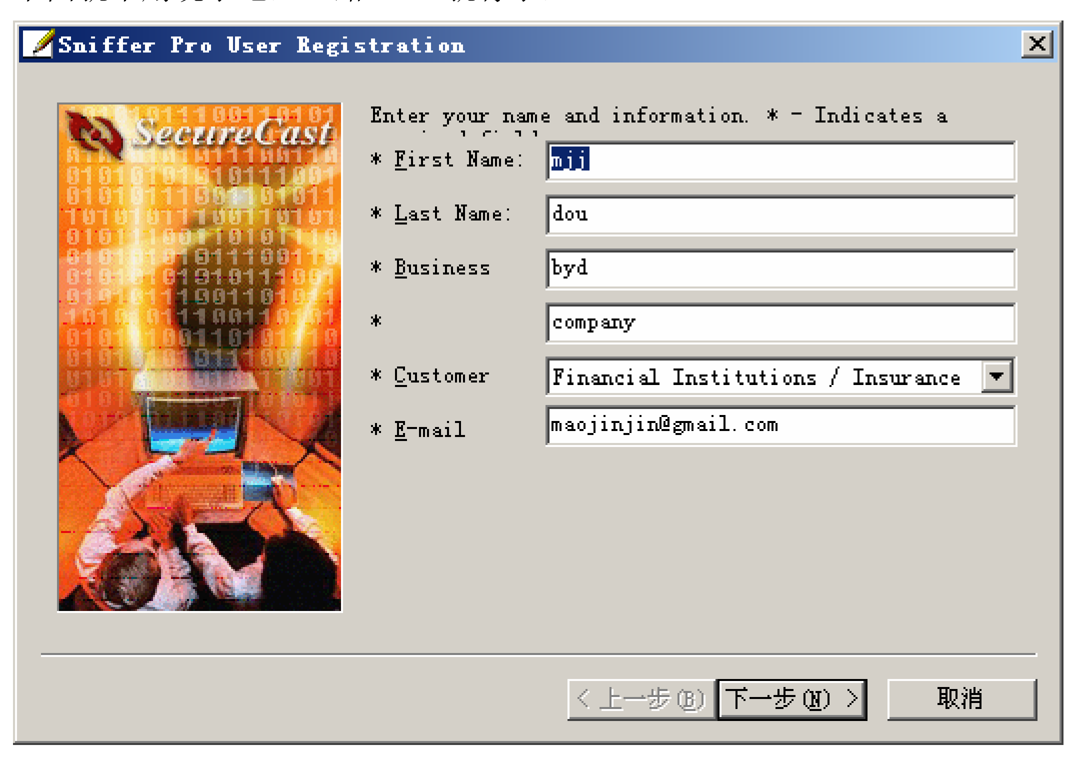
好了，开始我们激动人心的探索吧!!!

一、 sniffer 的安装

双击 sniffer 的安装程序。



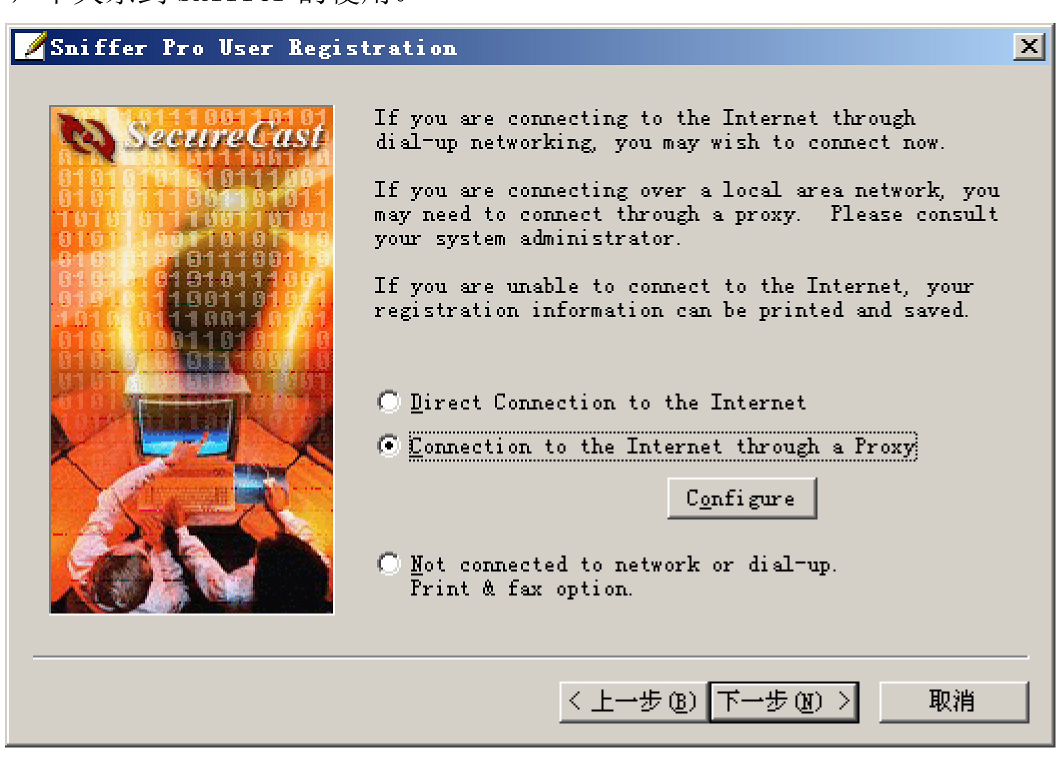
下面就不用说了吧，一路 NEXT 就行了，



在上面填入你的个人信息，记住有*号的一定要填上。



在这里的选项并不重要，最后一个 Sniffer Serial Number 才是关键，需要你填入产品序列号，第一个“Please let us know where you heard about this product”表示“你是通过什么途径知道 sniffer 的”，随便选一个就行了，第二个“Do you wish receive announcements about this product?”表示“你是否愿意用前面填写的信息接受来自 sniffer 方面的公告”当然示个人爱好而定，如果不想收到这方面的信息，选“NO”就行了，下面还有接收方式，如：E-mail，FAX 等，第三个“May we share your name with other companies that use NAI products?”表示“是否愿意与其它使用 NAI 公司产品的用户分享”，这个随便选择，不关系到 Sniffer 的使用。



上图就显的重要些了，在这里选择你接入 Internet 的方式，第一个方式是直接接入 Internet，第二个方式是通过代理接入 Internet，这种方式下面有个“Configure”按钮大家需要注意：

Proxy Communication Configuration

HTTP Proxy Setup

Use HTTP proxy at　　Port

User

Password:

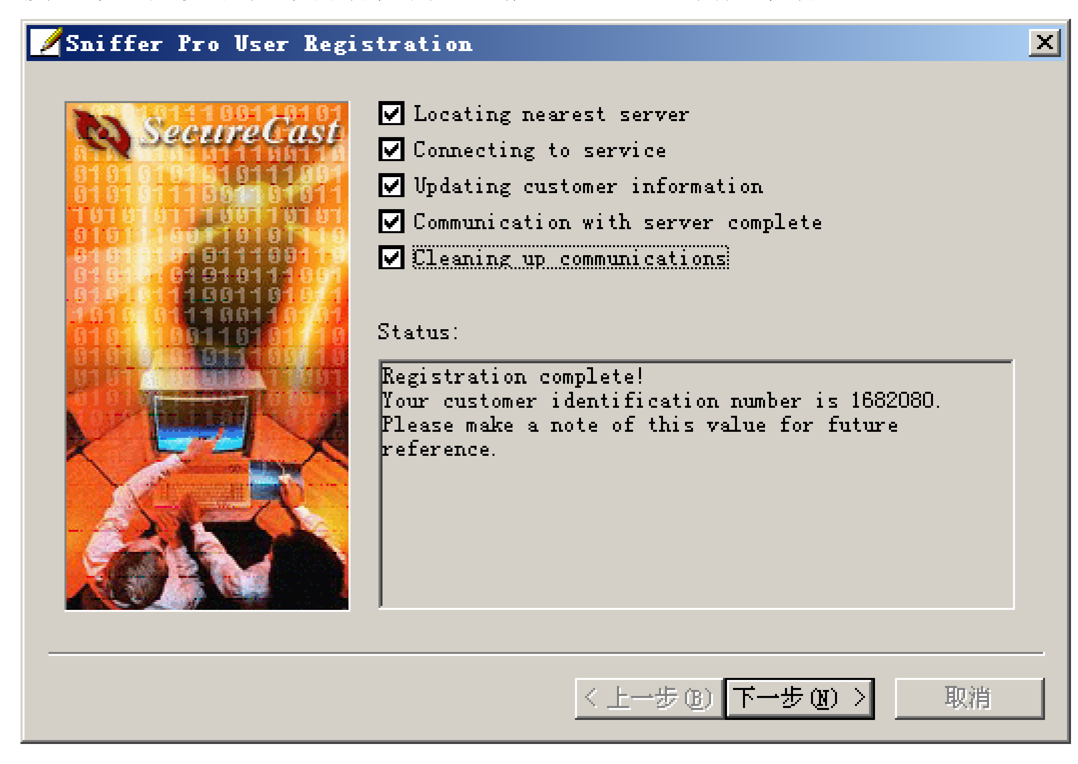
Your user name and password will not be stored.

OK　Cancel

在这里可以填入你的代理服务器及端口，如果需要用户名与密码你也必需填入，当然在 Sniffer 中，这些是不被软件存储的，需要你自已时时填入。(注意，在默认情况下，它会根据你浏览器的设置来自动填写)。

最后一个选项“Not connected to network or dial-up.Print & fax option”表示如果你没有接入 Internet，你的注册信息将被打印或传真表示出来，这里要根据每个人的具体情况来填写，我的就是选择“Connection to the Internet through a Proxy”通过代理接入，这表示你的 PC 可能是接在一个 LAN 当中。

接下来，是安装程序自动检测一些信息，这里不需要手动。

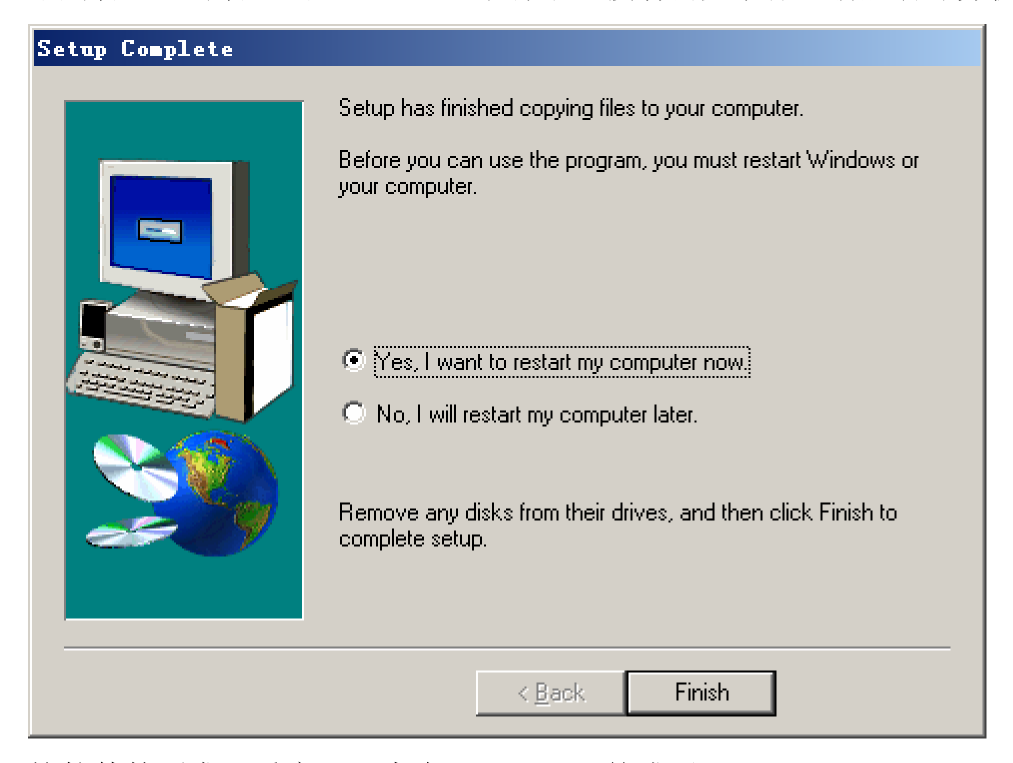


以上都是自动完成的。

这下面这张图里是你的个人注册信息，大家可以看到你前面所填写的很多信息。



不用管它，尽管点击“Finish”就行了。接着会要求你重新启动计算机。



按软件的要求，重启吧，点击“Finish”就成了！

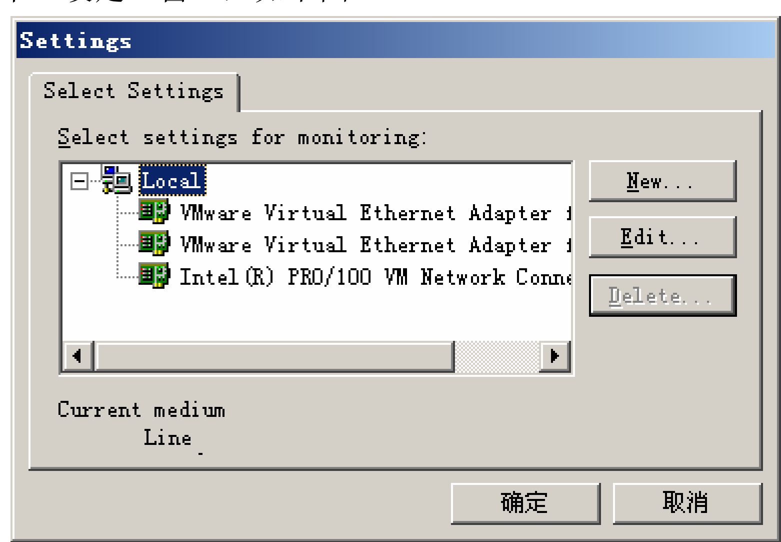
至此 Sniffer 安装完成了，大家敢快运行看看吧！

如果大家要汉化的话，直接运行汉化包就行了！

二、 Sniffer 的使用

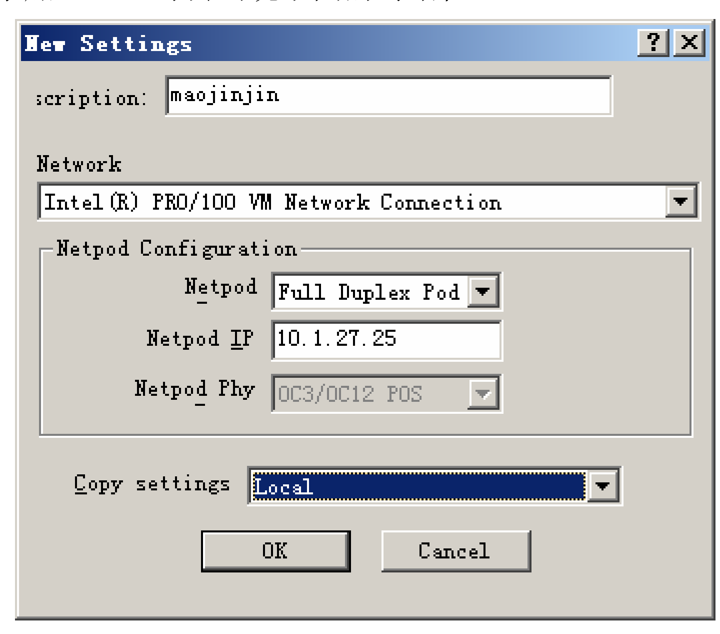
接入来我们就开始学习使用 Sniffer pro 及怎么配置它，如何实现它强大的功能。

第一次启动 Sniffer Pro 时，如果系统安装了多个网卡，系统会要求选择使用的代理或者网卡。结果选择储存在 Sniffer Pro 中，下次运行时就会自动选择同样的代理。可以在启动 Sniffer Pro 后从**“文件—选择设定”**中改变代理。这时会出现一个“设定”窗口，如下图



大家看到我有三块网卡，其中两块是我装 VMware Workstation 虚拟机时产生了，第三块 Inter(R)开头的那个才是真正的物理网卡。

如果想从另一块网卡上捕获流量数据，但是在设定窗口中没有显示这块网卡，单击 New（添加）。这时会出现下图的对话框。



输入以下信息：

■ **说明** 为适配器取一个好名字。在选择了 Select Settings（选择设定）选项时就会出现说明域。

■ **网络适配器**　这里将显示出你系统上所有 NDIS 3.1+相容网络适配器。选择一个没有被 Sniffer Pro 探针监控的适配器。

■ **类型**　说明探针是远程的还是本地的。Sniffer Pro 限定只能使用本地探针。只有分布式 Sniffer 才支持远程探针。

■ **复制设定**　从当前代理中复制配置设定。这个下拉式列表显示了你系统上以前定义过的所有代理。

Sniffer Pro 允许同时在多个区段捕获数据。要做到这一点，就要先启动一个 Sniffer Pro 对话，为要监控的第一个区段选择代理，并在那个接口上开始捕获。然后启动一个新的 Sniffer Pro 对话，选择下一个代理，并开始捕获。可以把这两个窗口平铺在屏幕上，这样可以同时观察到这些区段的情况。在想要在一个区段同时进行捕获与监控时，也可以使用这种方法。

同时从网络上的两个不同区段捕获流量的能力非常有用。例如，如果有一台客户端电脑位于某个区段上，一台服务器位于另一个区段，就可以同时捕获这两个区段的流量。

NetPod

以太网可以以全双工模式工作，每个方向的传输与接收流量速度都可以达到 100Mbps。如果只使用 100Mbps 的接收通道，那么以 200Mbps 的速度捕获数据就会出现问题。Network Associates 公司的高速以太网全双工 Pod 是一种硬件设备，可以用它来捕获全双工流量。它可以捕获网络上的数据，并储存在一个中间缓冲设备中。接着被捕获的数据通过另一个高速以太网连接，被传递给 Sniffer Pro。Sniffer Pro 系统中必须有一个得到系统支持的高速以太网适配器。得到系统支持的网卡包括：

■ 带有 NAI 增强驱动程序的 Adaptec Cogent 10/100 高速以太网 21140UC PCI 适配器。

■ 带有 NAI 增强驱动程序的 Network Associates NAI2140/UC PCI 高速以太网适配器。

■ 带有 NAI 增强驱动程序的 Xircom CBE2－100BTX CardBus 适配器。

n 带有 NAI 增强驱动程序的 Network Associates CardBus 以太网二代 10/100 适配器。

如果要获得更多关于高速以太网全双工 Pod 的信息，或者要购买，可以与当地 NAI 代表或者代销商联系。

如果要在 Sniffer Pro 中设置高速以太网 Pod，可以选择文件一选择设定。单击 New（添加）按钮来定义与高速以太网全双工 Pod 相适应的新的本地代理。在 Network Adapter（网卡）中，选择与高速以太网全双工 Pod 相适应的网卡。从 NetPod Type（NetPod 类型）中，选择 Full Duplex Pod（全双工 Pod）。在 NetPod IP Address（NetPod IP 地址）中，输入 Sniffer Pro 系统的网络适配器的 IP 地址再加 1。例如，如果 Sniffer Pro 的网络适配器地址为 192.168.1.1，就必须设定 192.168.1.2 为 NetPod IP 地址。如果 NetPod 要正常工作，Sniffer Pro 系统必须安装 TCP/IP。全双工 Pod 要求有静态 IP 地址，所以应该禁用 DHCP。单击 OK 按钮完成代理安装。设定好全双工 Pod 并选择代理后，就可以捕获数据了。

Sniffer Pro 的当前版本只支持对全双工 Pod 的本地连接。不能通过网络来连接 Pod。

下面我们来看看如何在实际环境中使用 Sniffer.

一、 捕获数据包前的准备工作

在默认情况下，sniffer 将捕获其接入碰撞域中流经的所有数据包，但在某些场景下，有些数据包可能不是我们所需要的，为了快速定位网络问题所在，有必要对所要捕获的数据包作过滤。Sniffer 提供了捕获数据包前的过滤规则的定义，过滤规则包括 2、3 层地址的定义和几百种协议的定义。定义过滤规则的做法一般如下：

1、在主界面选择 capture→define filter 选项。

2、define filter→address，这是最常用的定义。其中包括 MAC 地址、ip 地址和 ipx 地址的定义。以定义 IP 地址过滤为例，见图 1。

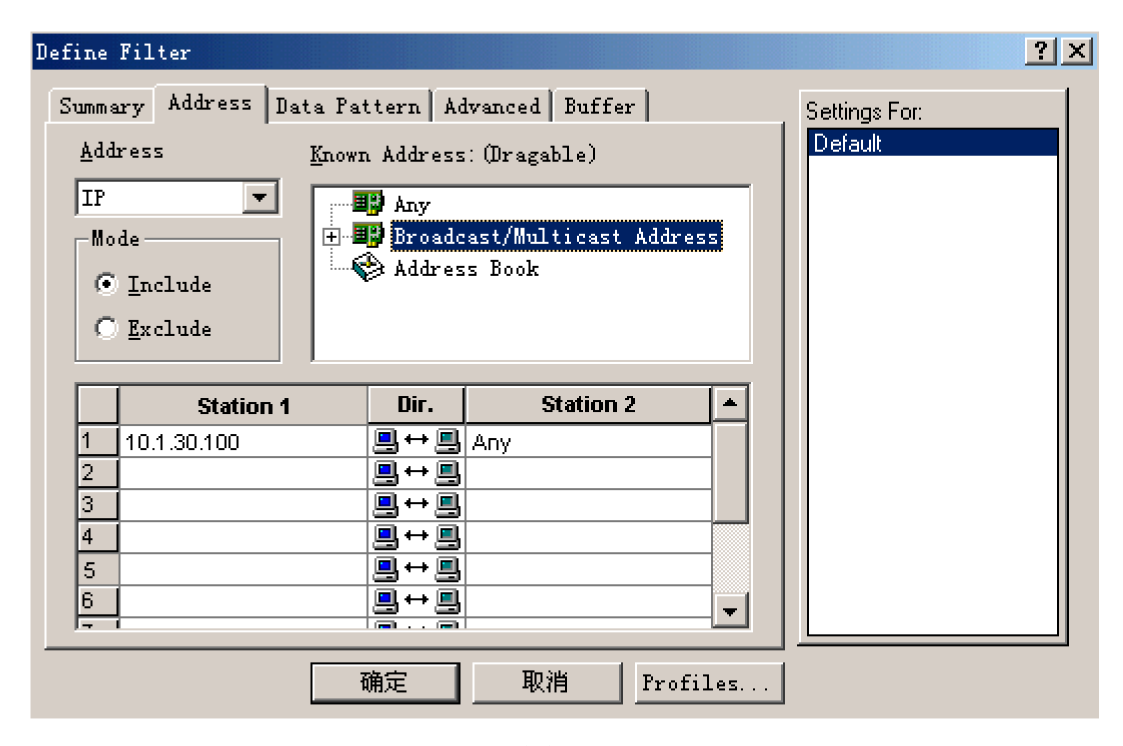


图 1

比如，现在要捕获地址为 10.1.30.100 的主机与其他主机通信的信息，在 Mode 选项卡中，选 Include(选 Exclude 选项，是表示捕获除此地址外所有的数据包)；在 station 选项中，在任意一栏填上 10.1.30.100，另外一栏填上 any(any 表示所有的 IP 地址）。这样就完成了地址的定义。

注意到 Dir. 栏的图标：



表示，捕获 station1 收发的数据包；



表示，捕获 station1 发送的数据包；



表示，捕获 station1 收到的数据包。

最后，选取

Profiles...

，将定义的规则保存下来，供以后使用。

3、Define filter→advanced，定义希望捕获的相关协议的数据包。如图 2。

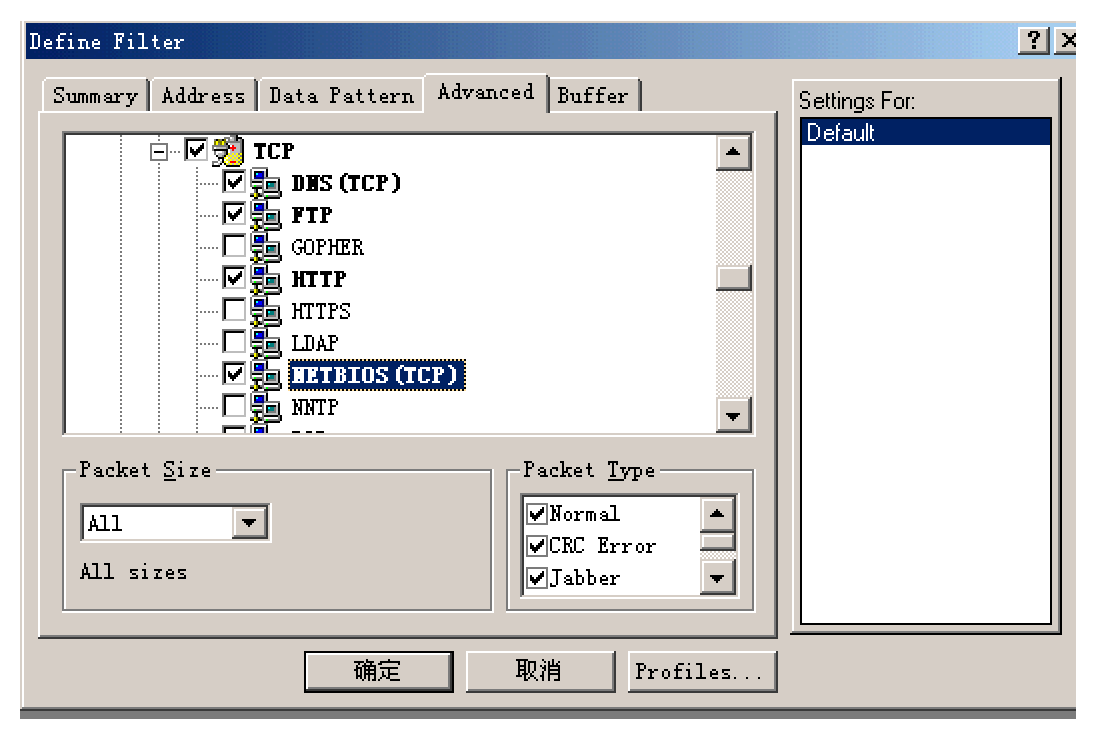


图 2

比如，想捕获 FTP、NETBIOS、DNS、HTTP 的数据包，那么说首先打开 TCP 选项卡，再进一步选协议；还要明确 DNS、NETBIOS 的数据包有些是属于 UDP 协议，故需在 UDP 选项卡做类似 TCP 选项卡的工作，否则捕获的数据包将不全。

如果不选任何协议，则捕获所有协议的数据包。

PacketSize 选项中，可以定义捕获的包大小，图 3，是定义捕获包大小界于 64 至 128bytes 的数据包。

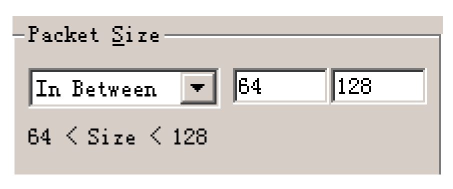


图 3

4、definefilter→buffer,定义捕获数据包的缓冲区。如图 4：

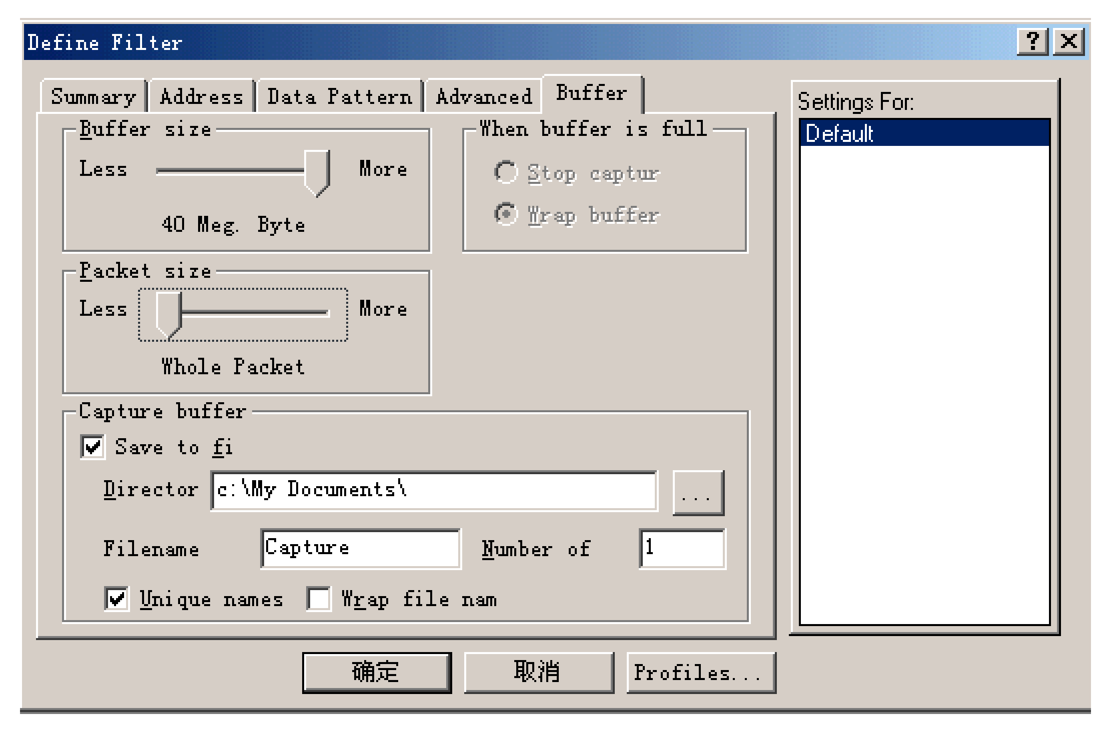


图 4

Buffersize 选项卡，将其设为最大 40M。

Capture buffer 选项卡，将设置缓冲区文件存放的位置。

5、最后，需将定义的过滤规则应用于捕获中。如图 5：

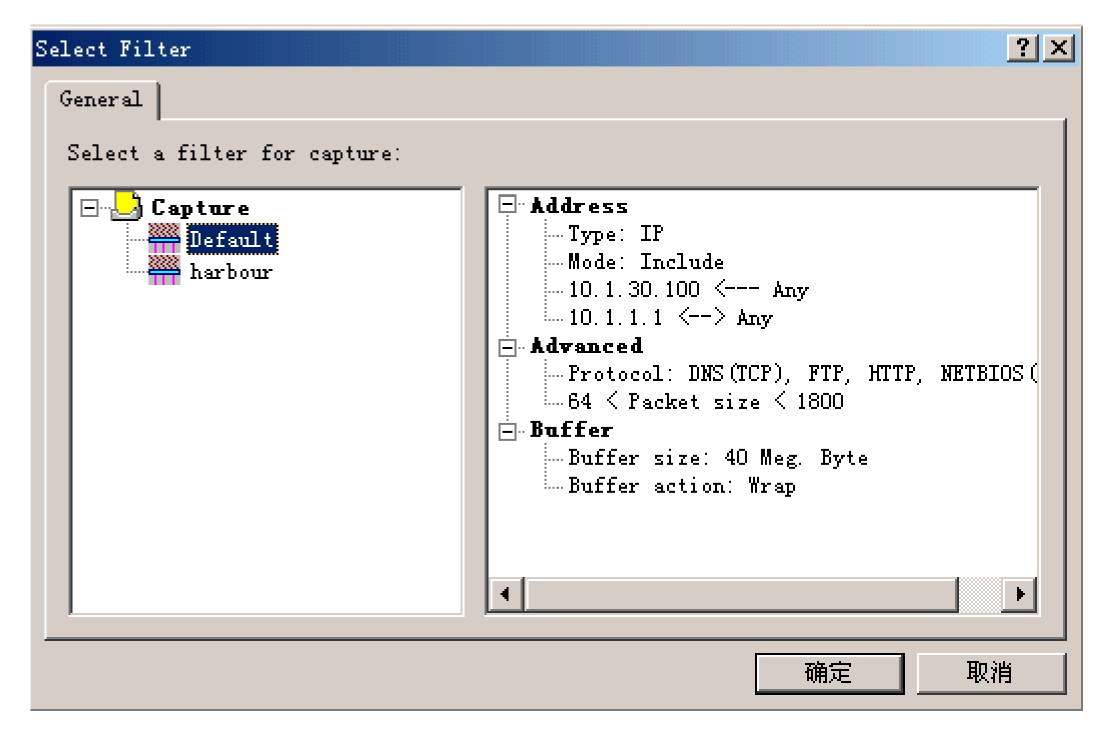


图 5

点选 SelectFilter→Capture 中选取定义的捕获规则。

二、 捕获数据包时观察到的信息

Capture→Start,启动捕获引擎。

sniffer 可以实时监控主机、协议、应用程序、不同包类型等的分布情况。

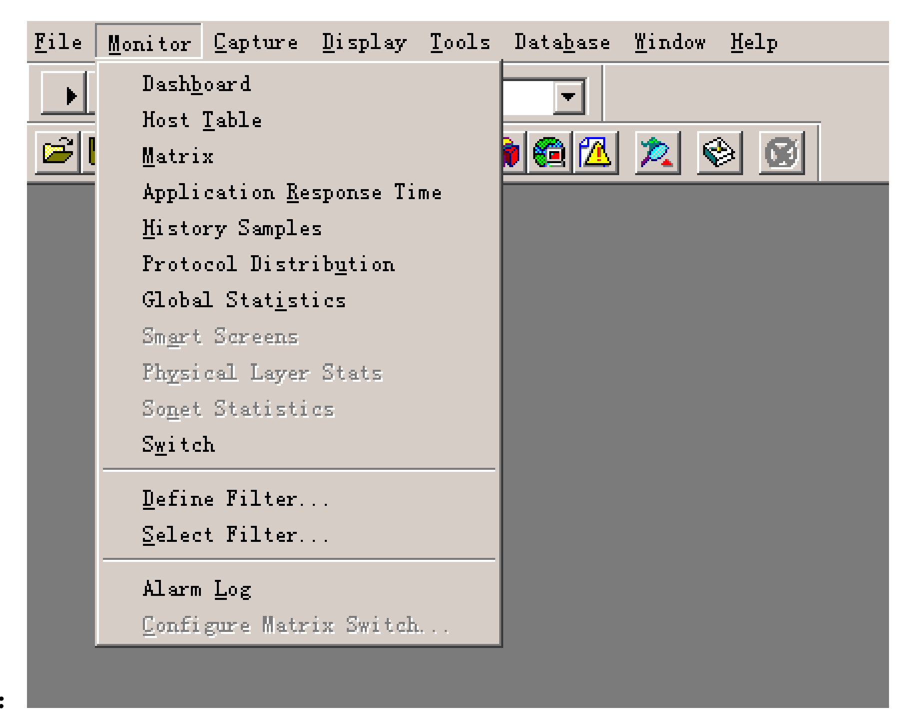
如图 6：

图 6

Dashboard:可以实时统计每秒钟接收到的包的数量、出错包的数量、丢弃包的数量、广播包的数量、多播包的数量以及带宽的利用率等。

HostTable：可以查看通信量最大的前 10 位主机。

Matrix:通过连线，可以形象的看到不同主机之间的通信。

ApplicationResponseTime:可以了解到不同主机通信的最小、最大、平均响应时间方面的信息。

HistorySamples:可以看到历史数据抽样出来的统计值。

Protocoldistribution:可以实时观察到数据流中不同协议的分布情况。

Switch:可以获取 cisco 交换机的状态信息。

在捕获过程中，同样可以对想观察的信息定义过滤规则，操作方式类似捕获前的过滤规则。

三、 捕获数据包后的分析工作

要停止 sniffer 捕获包时，点选 Capture→Stop 或者 Capture→Stop and Display,前者停止捕获包，后者停止捕获包并把捕获的数据包进行解码和显示。如图 7：

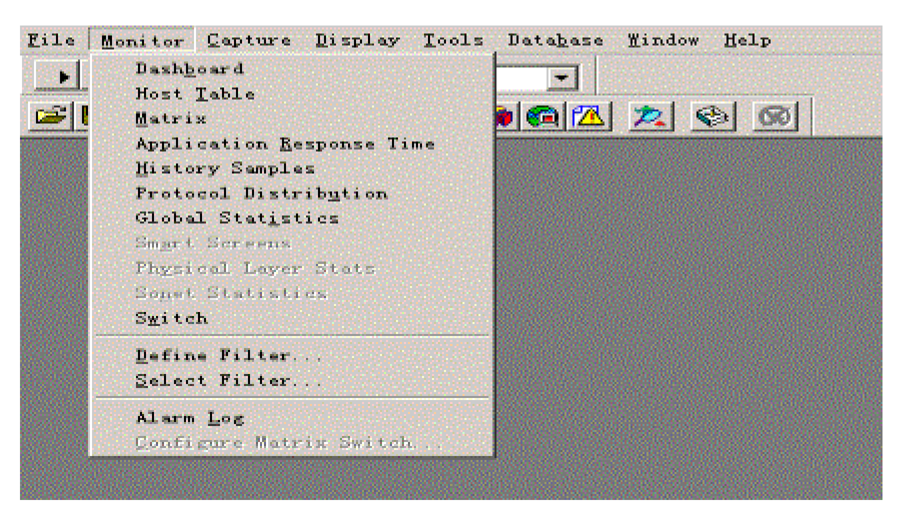


图 7

Decode:对每个数据包进行解码，可以看到整个包的结构及从链路层到应用层的信息，事实上，sniffer 的使用中大部分的时间都花费在这上面的分析，同时也对使用者在网络的理论及实践经验上提出较高的要求。素质较高的使用者借此工具便可看穿网络问题的结症所在。

Expert:这是 sniffer 提供的专家模式，系统自身根据捕获的数据包从链路层到应用层进行分类并作出诊断。其中 diagnoses 提出非常有价值的诊断信息。图 8，是 sniffer 侦查到 IP 地址重叠的例子及相关的解析。

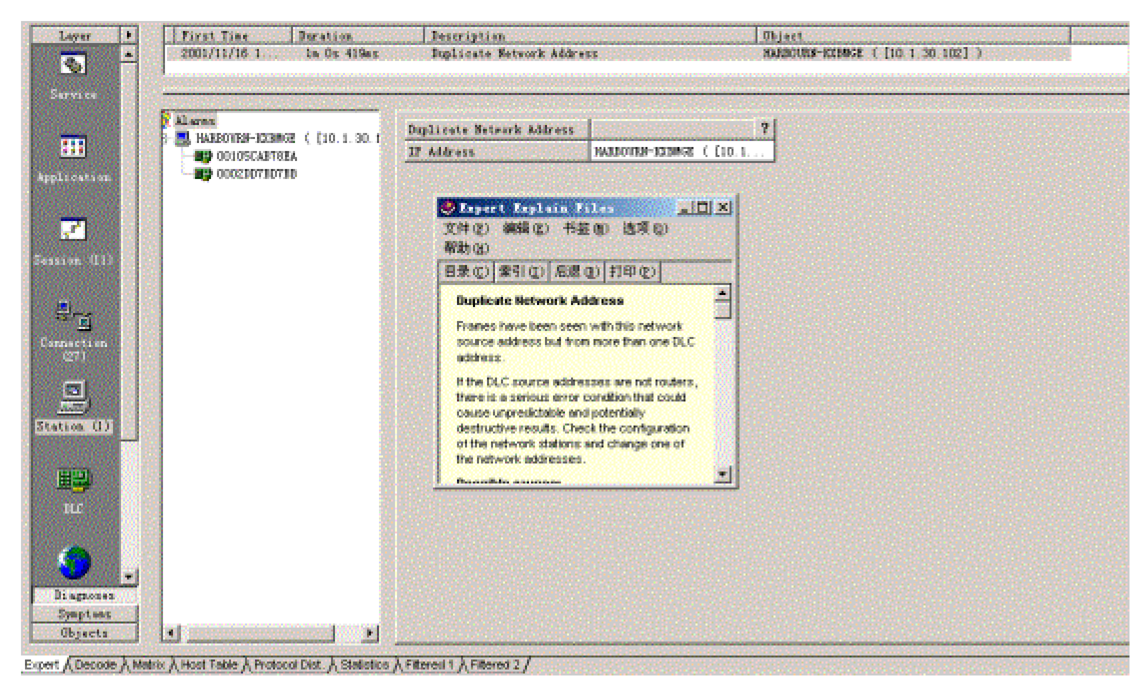


图 8

sniffer 同样提供解码后的数据包过滤显示。

要对包进行显示过滤需切换到 Decode 模式。

Display—〉 definefilter，定义过滤规则。

Display—〉 selectfilter,应用过滤规则。

显示过滤的使用基本上跟捕获过滤的使用相同。

四、 sniffer 提供的工具应用

sniffer 除了提供数据包的捕获、解码及诊断外，还提供了一系列的工具，包括包发生器、ping、traceroute、DNSlookup、finger、whois 等工具。

其中，包发生器比较有特色，将做简单介绍。其他工具在操作系统中也有提供，不做介绍。

包发生器提供三种生成数据包的方式：

点选



，新构一个数据包，包头、包内容及包长由用户直接填写。图 9，定义一个广播包，使其连续发送，包的发送延迟位 1ms

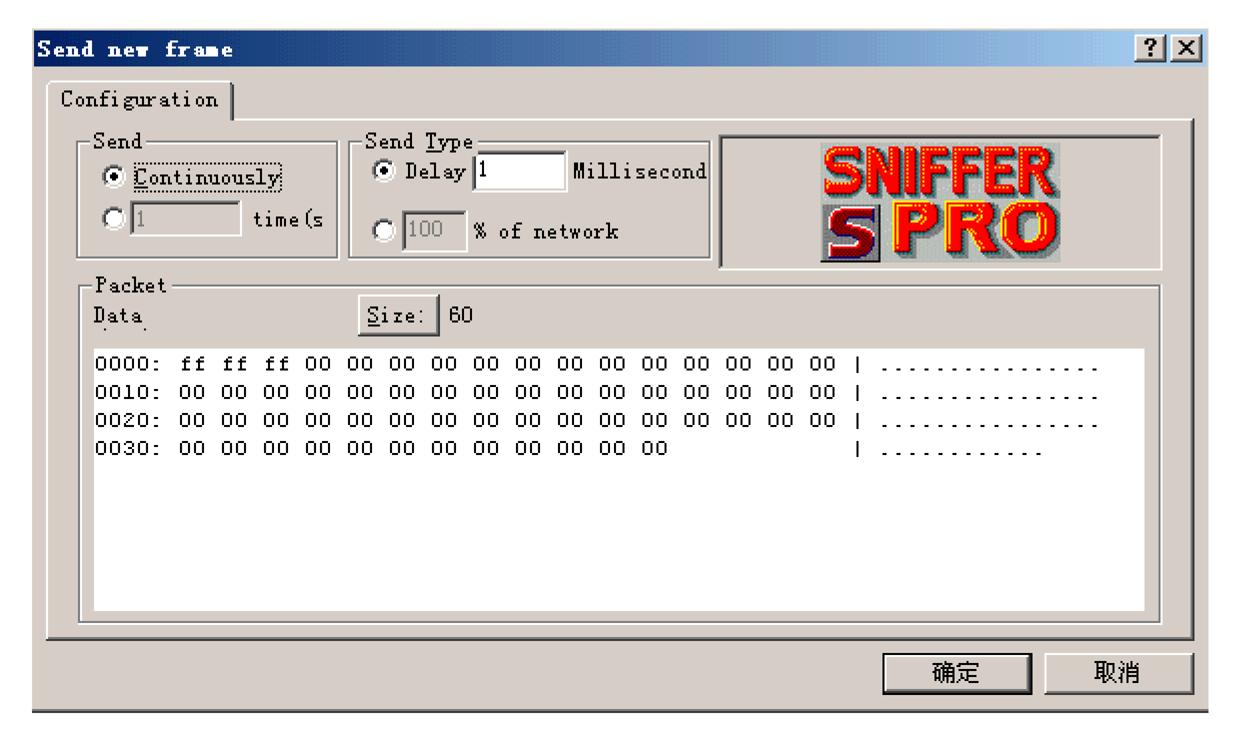


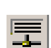
图 9

点选



发送在 Decode 中所定位的数据包，同时可以在此包的基础上对数据包进行如前述的修改。

点选



，发送 buffer 中所有的数据包，实现数据流的重放。见图 10：

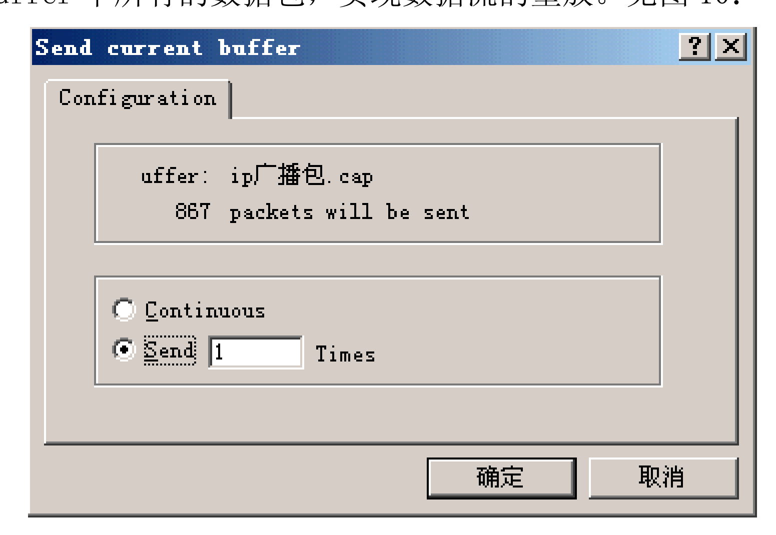


图 10

可以定义连续地发送 buffer 中地数据包或只发送一次 buffer 中地数据包。请特别注意，不要在运行的网络中重放数据包，否则容易引起严重的网络问题。数据包的重放经常用于实验环境中。

Sniffer 使用简介（二）

前面一章中我们从大体上介绍了如何使用 sniffer，下面我们将继续讲解 sniffer 的一些使用及一些常见面板的含义。

Sniffer Pro的基本使用和实例

运行环境及安装

Sniffer Pro 可运行在局域网的任何一台机器上，如果是练习使用，网络连接最好用 Hub 且在一个子网，这样能抓到连到 Hub 上每台机器传输的包。

本文用的版本是 4.75 sniffer Pro软件的获取可在www.baidu.com或 www.google.com 中输入Sniffer Pro 4.75，查找相应的下载站点来下载。(当然你也可以用Sniffer 4.6 的， 该版本是不要序列号的)

安装非常简单，setup 后一路确定即可，第一次运行时需要选择你的网卡。

最好在 win2000 下运行，在 win2003 下运行网络流量表有问题。

常用功能介绍

1、Dashboard （网络流量表）

点击图 1 中①所指的图标，出现三个表，第一个表显示的是网络的使用率（Utilization），第二个表显示的是网络的每秒钟通过的包数量（Packets），第三个表显示的是网络的每秒错误率（Errors）。通过这三个表可以直观的观察到网络的使用情况，红色部分显示的是根据网络要求设置的上限。

选择图 1 中②所指的选项将显示如图 2 所示的更为详细的网络相关数据的曲线图。每个子项的含义无需多言，下面介绍一下测试网络速度中的几个常用单位。

在 TCP/IP 协议中，数据被分成若干个包（Packets）进行传输，包的大小跟操作系统和网络带宽都有关系，一般为 64、128、256、512、1024、1460 等，包的单位是字节。

很多初学者对 Kbps、KB、Mbps 等单位不太明白，B 和 b 分别代表 Bytes(字节) 和 bits（比特），1 比特就是 0 或 1。1 Byte = 8 bits 。1Mbps (megabits per second 兆比特每秒)，亦即 1 x 1024 / 8 = 128KB/sec（字节/秒），我们常用的 ADSL 下行 512K 指的是每秒 512K 比特(Kb)， 也就是每秒 512/8=64K 字节（KB）

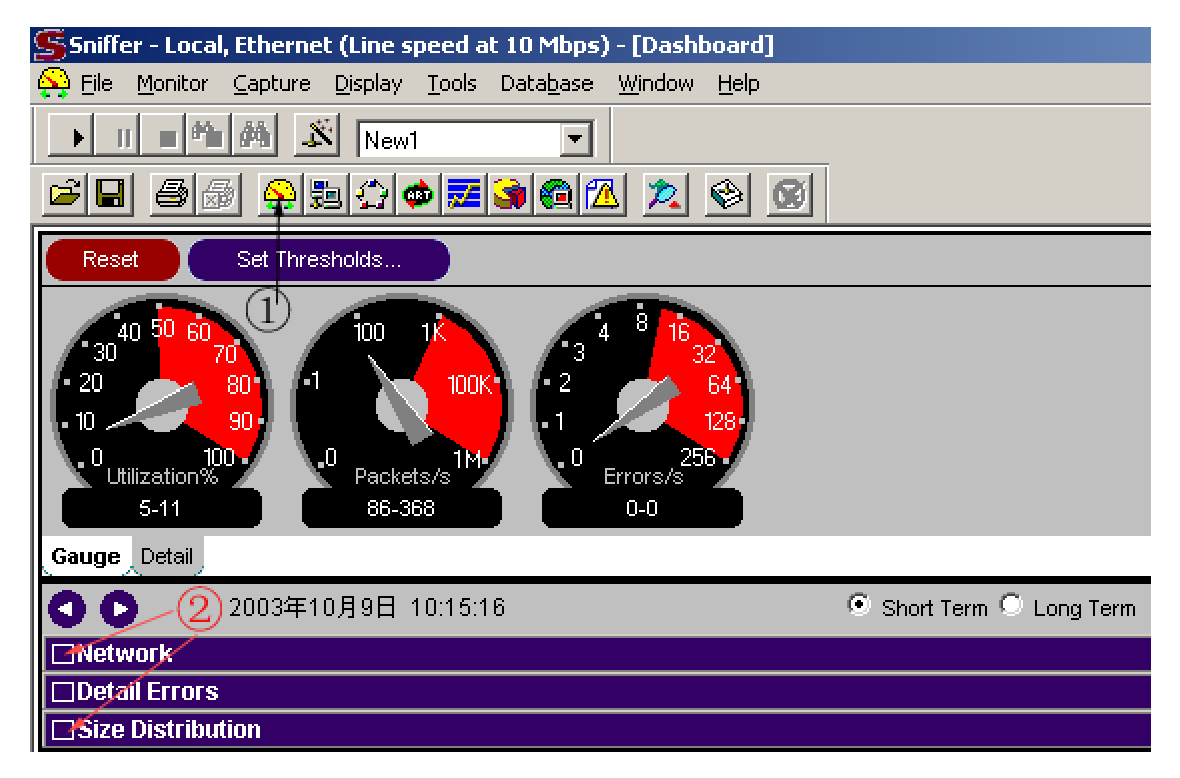


图 1

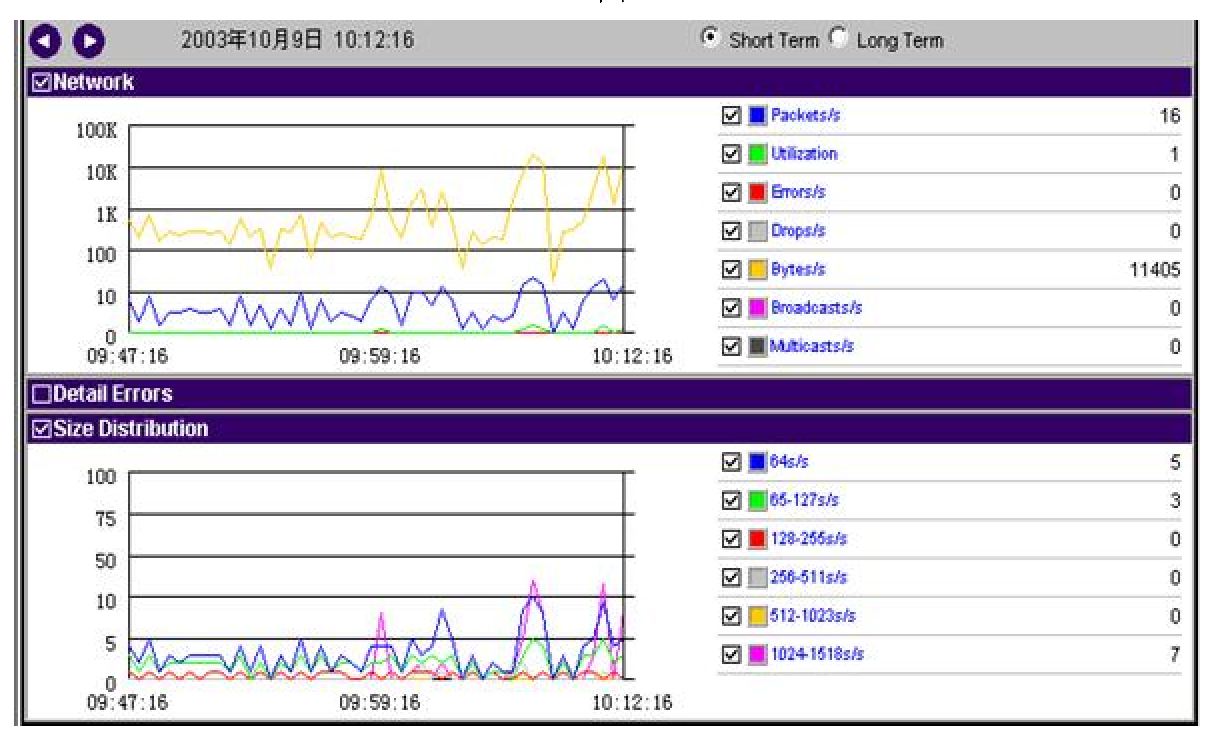


图 2

2、Host table（主机列表）

如图 3 所示，点击图 3 中①所指的图标，出现图中显示的界面，选择图中②所指的 IP 选项，界面中出现的是所有在线的本网主机地址及连到外网的外网服

务器地址，此时想看看 192.168.113.88 这台机器的上网情况，只需如图中③所示单击该地址出现图 4 界面。

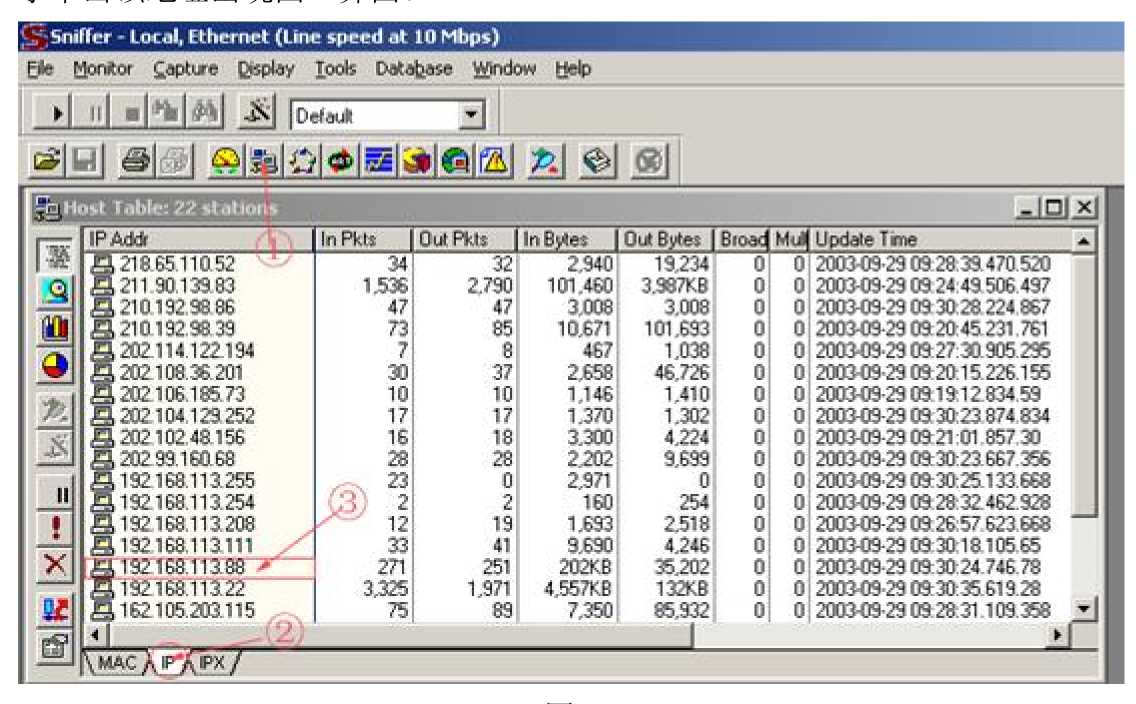


图 3

图 4 中清楚地显示出该机器连接的地址。点击左栏中其它的图标都会弹出该机器连接情况的相关数据的界面。

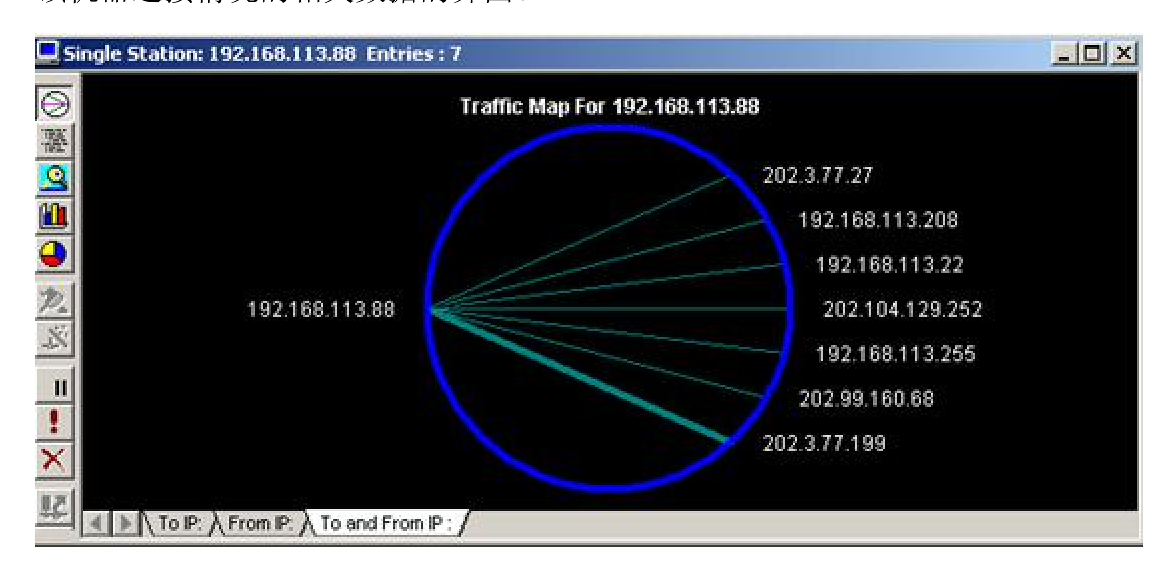
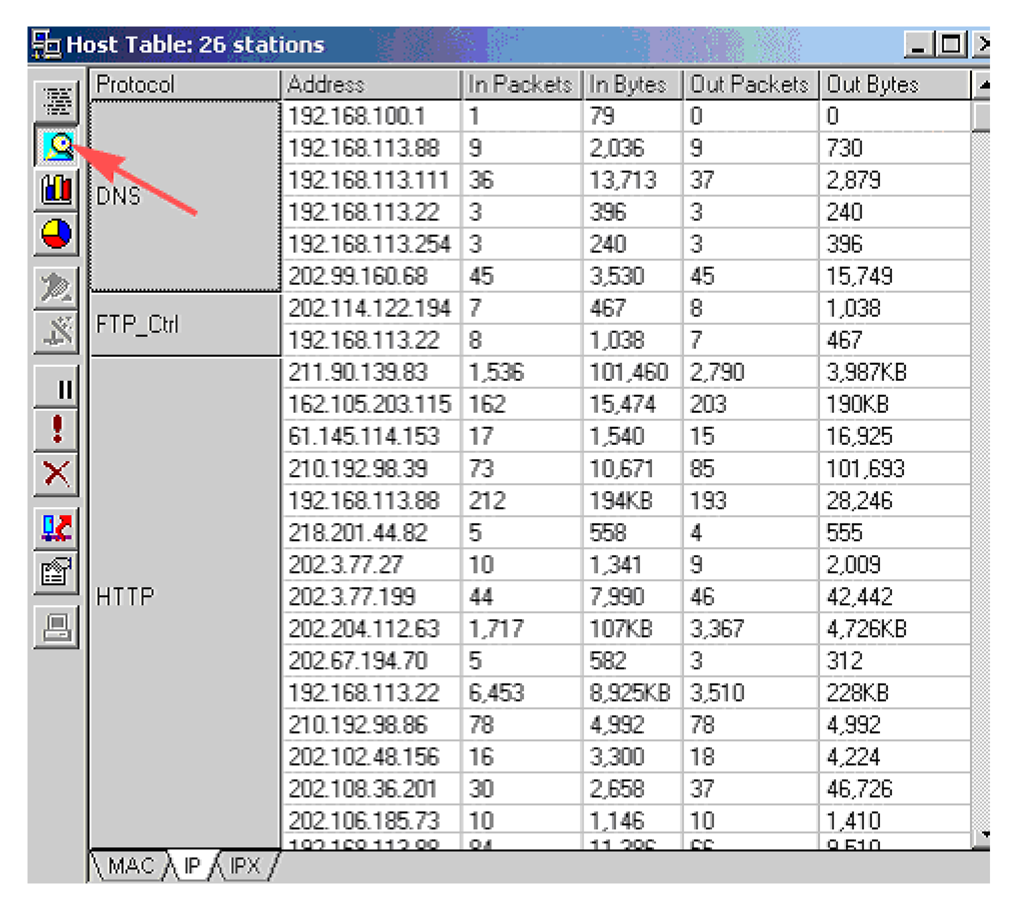


图 4

3、Detail（协议列表）

点击图 5 所示的“Detail”图标，图中显示的是整个网络中的协议分布情况，可清楚地看出哪台机器运行了那些协议。注意，此时是在图 3 的界面上点击的，如果在图 4 的界面上点击显示的是那台机器的情况。



Host Table: 26 stations

Protocol	Address	In Packets	In Bytes	Out Packets	Out Bytes
DNS	192.168.100.1	1	79	0	0
	192.168.113.88	9	2,036	9	730
	192.168.113.111	36	13,713	37	2,879
	192.168.113.22	3	396	3	240
	192.168.113.254	3	240	3	396
	202.99.160.68	45	3,530	45	15,749
FTP_Ctrl	202.114.122.194	7	467	8	1,038
	192.168.113.22	8	1,038	7	467
HTTP	211.90.139.83	1,536	101,460	2,790	3,987KB
	162.105.203.115	162	15,474	203	190KB
	61.145.114.153	17	1,540	15	16,925
	210.192.98.39	73	10,671	85	101,693
	192.168.113.88	212	194KB	193	28,246
	218.201.44.82	5	558	4	555
	202.3.77.27	10	1,341	9	2,009
	202.3.77.199	44	7,990	46	42,442
	202.204.112.63	1,717	107KB	3,367	4,726KB
	202.67.194.70	5	582	3	312
	192.168.113.22	6,453	8,925KB	3,510	228KB
	210.192.98.86	78	4,992	78	4,992
	202.102.48.156	16	3,300	18	4,224
	202.108.36.201	30	2,658	37	46,726
	202.106.185.73	10	1,146	10	1,410

图 5

4、Bar（流量列表）

点击图 6 所示的“Bar”图标，图中显示的是整个网络中的机器所用带宽前 10 名的情况。显示方式是柱状图，图 7 显示的内容与图 6 相同，只是显示方式是饼图。

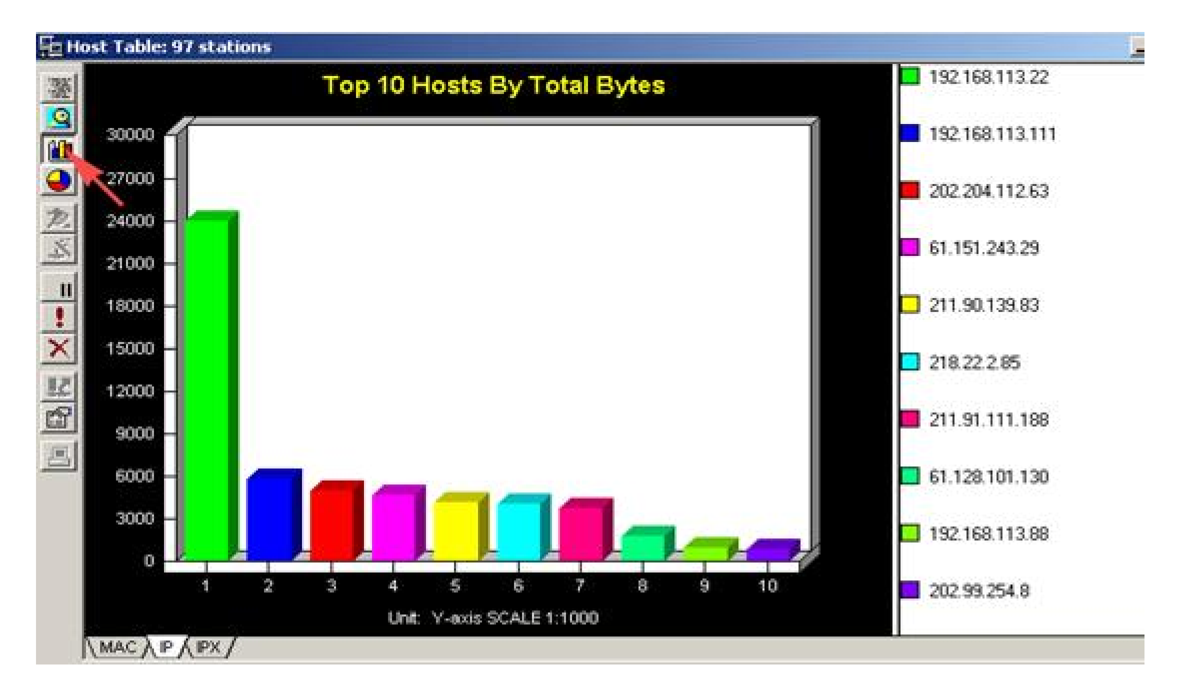


图 6

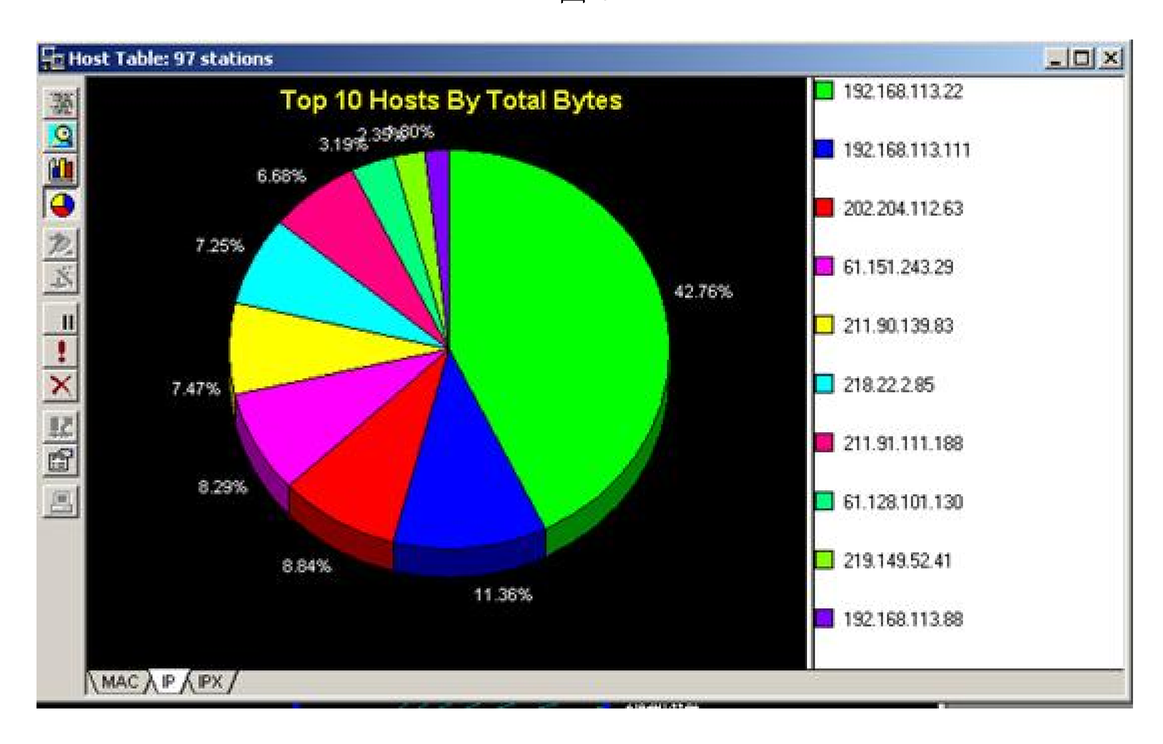


图 7

5、Matrix （网络连接）

点击图 8 中箭头所指的图标，出现全网的连接示意图，图中绿线表示正在发生的网络连接，蓝线表示过去发生的连接。将鼠标放到线上可以看出连接情况。鼠标右键在弹出的菜单中可选择放大（zoom）此图。

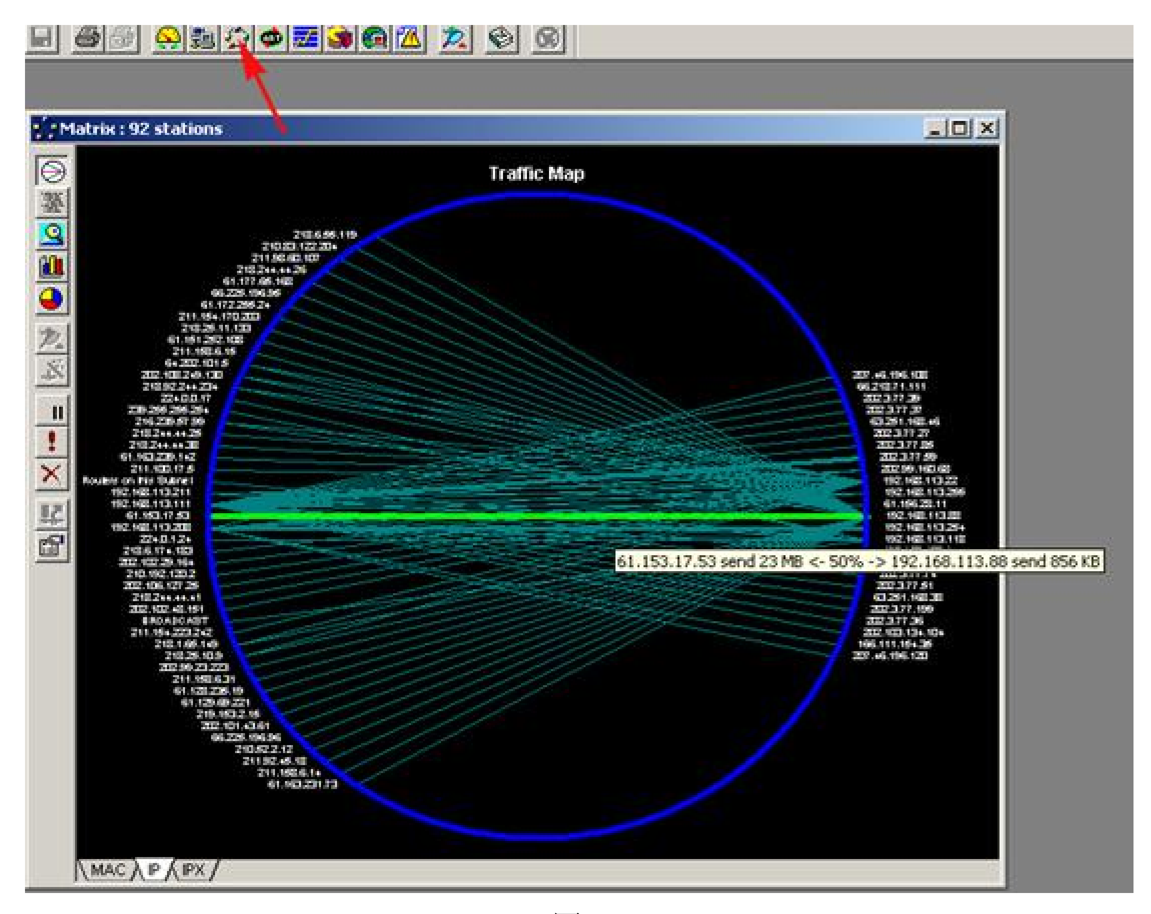


图 8

抓包实例

1、抓某台机器的所有数据包

如图 9 所示，本例要抓 192.168.113.208 这台机器的所有数据包，如图中①选择这台机器。点击②所指图标，出现图 10 界面，等到图 10 中箭头所指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 11 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。

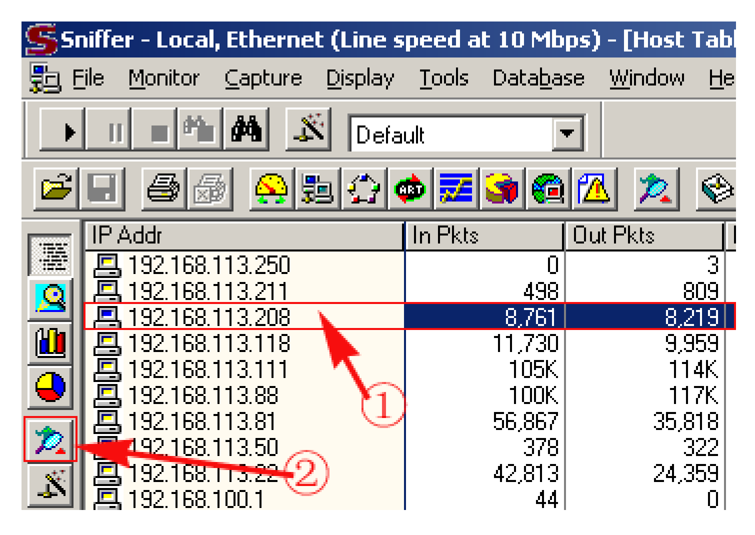


图 9

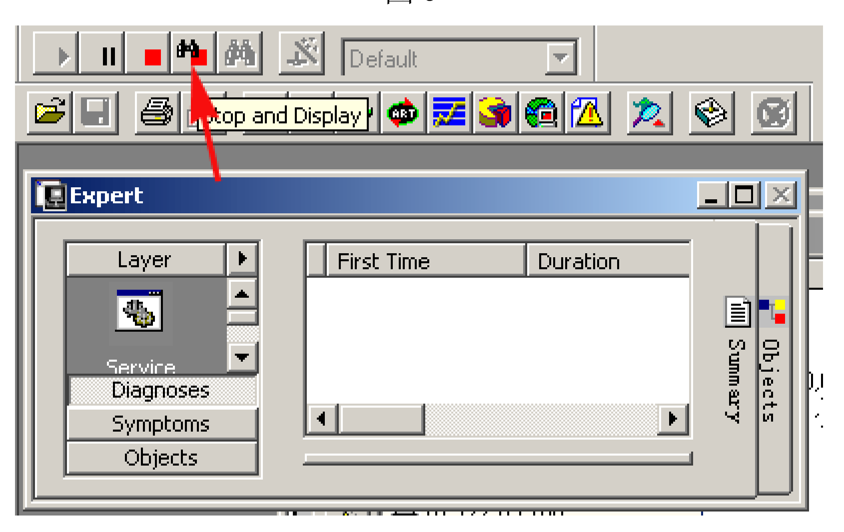


图 10

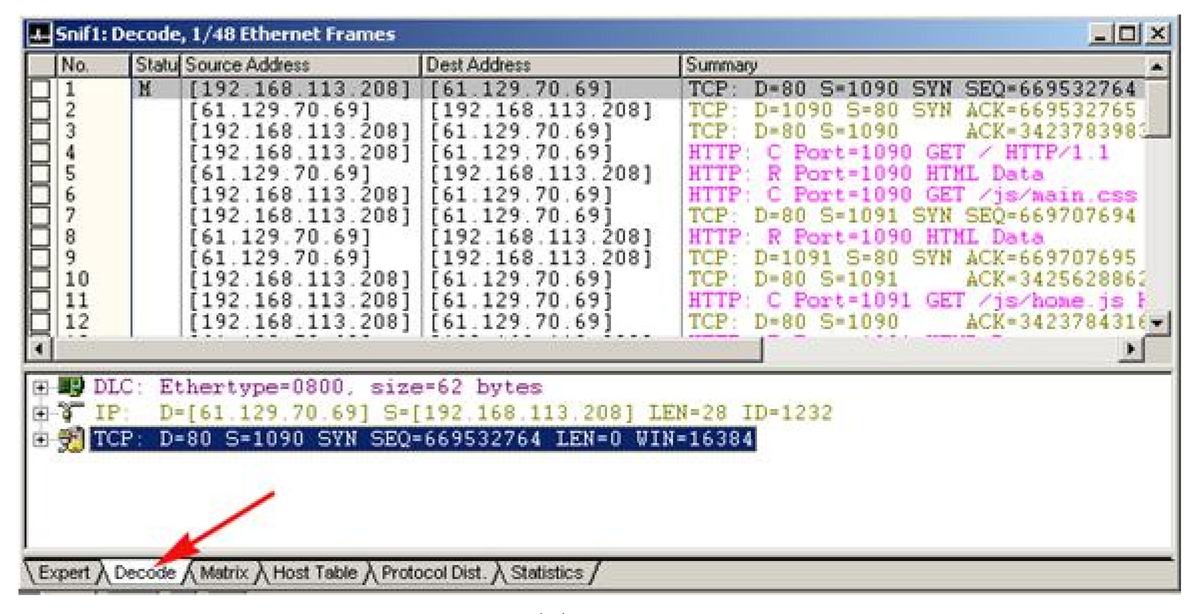


图 11

2、抓 Telnet 密码

本例从 192.168.113.208 这台机器 telnet 到 192.168.113.50，用 Sniff Pro 抓到用户名和密码。

步骤 1：设置规则

如图 12 所示，选择 Capture 菜单中的 Defind Filter，出现图 13 界面，选择图 13 中的 ADDress 项，在 station1 和 station2 中分别填写两台机器的 IP 地址，如图 14 所示选择 Advanced 选项，选择选 IP/TCP/Telnet，将 Packet Size 设置为 Equal 55， Packet Type 设置为 Normal.。

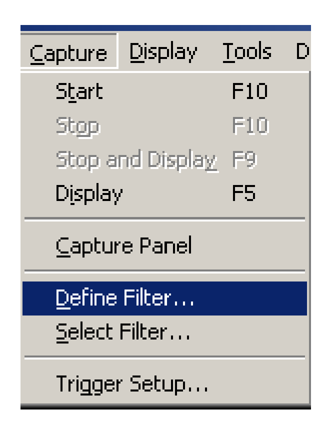

Capture
Display
Tools
D
Start F10
Stop F10
Stop and Display F9
Display F5
Capture Panel
Define Filter...
Select Filter...
Trigger Setup...

图 12



图 13

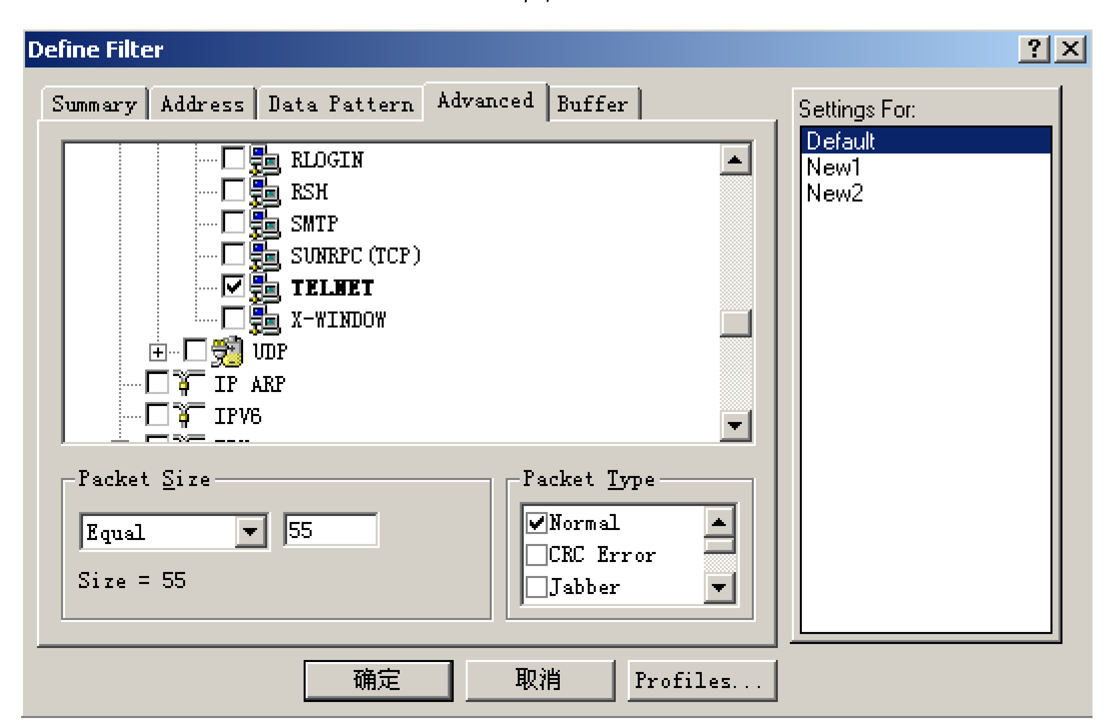


图 14

步骤 2：抓包

按 F10 键出现图 15 界面，开始抓包。

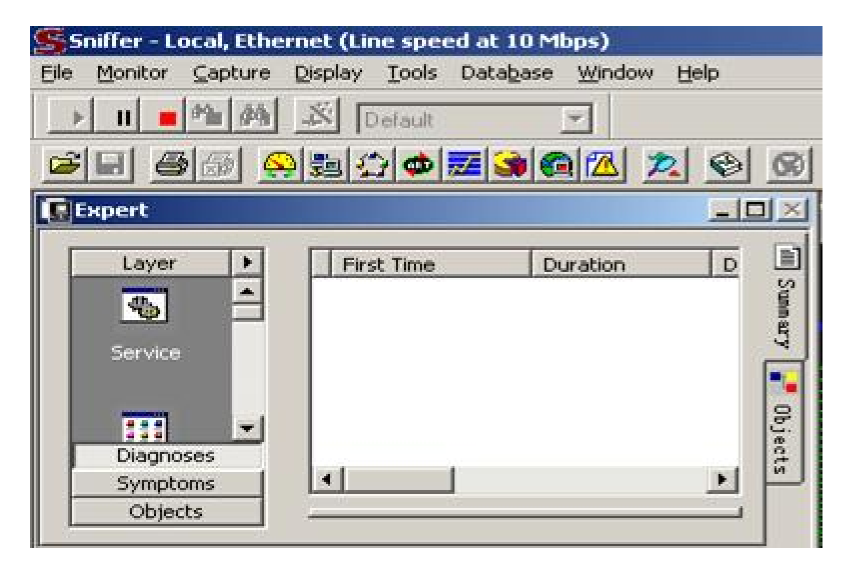


图 15

步骤 3：运行 telnet 命令

本例使 telnet 到一台开有 telnet 服务的 Linux 机器上。

telnet 192.168.113.50

login: test

Password:

步骤 4：察看结果

图 16 中箭头所指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 17 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。可以清楚地看出用户名为 test 密码为 123456。

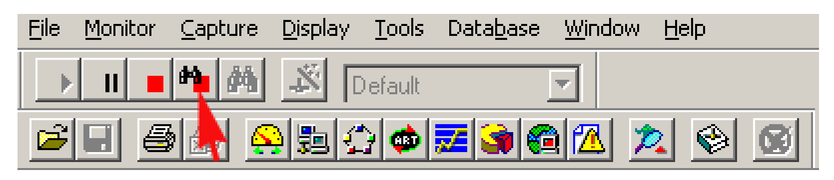


图 16

Snif2: Decode, 22/32 Ethernet Frames

No.	Status	Source Address	Dest Address	Summary
15		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2059 q
16		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2059 t
17		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2059 u
18		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2059 i
19		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2059 <08>
20		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2059 <08>
21		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2059 <08>
22		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 t
23		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 e
24		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 s
25		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 t
26		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 1
27		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 2
28		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 3
29		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 4
30		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 5
31		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 6
32		[192.168.113.208]	[192.168.113.50]	Telnet: C PORT=2060 <0D>

DLC: Ethertype=0800, size=60 bytes
IP: D=[192.168.113.50] S=[192.168.113.208] LEN=21 ID=24583
TCP: D=23 S=2060 ACK=859603127 SEQ=3488832343 LEN=1 WIN=17412
Telnet: C PORT=2060 t

Expert | Decode | Matrix | Host Table | Protocol Dist. | Statistics

图 17

解释：

虽然把密码抓到了，但大家也许对包大小（Packet Size）设为 55 不理解，网上的数据传送是把数据分成若干个包来传送，根据协议的不同包的大小也不相同，从图 18 可以看出当客户端 telnet 到服务端时一次只传送一个字节的数据，由于协议的头长度是一定的，所以 telnet 的数据包大小=DLC(14 字节)+IP(20 字节)+TCP(20 字节)+数据（一个字节）=55 字节，这样将 Packet Size 设为 55 正好能抓到用户名和密码，否则将抓到许多不相关的包。

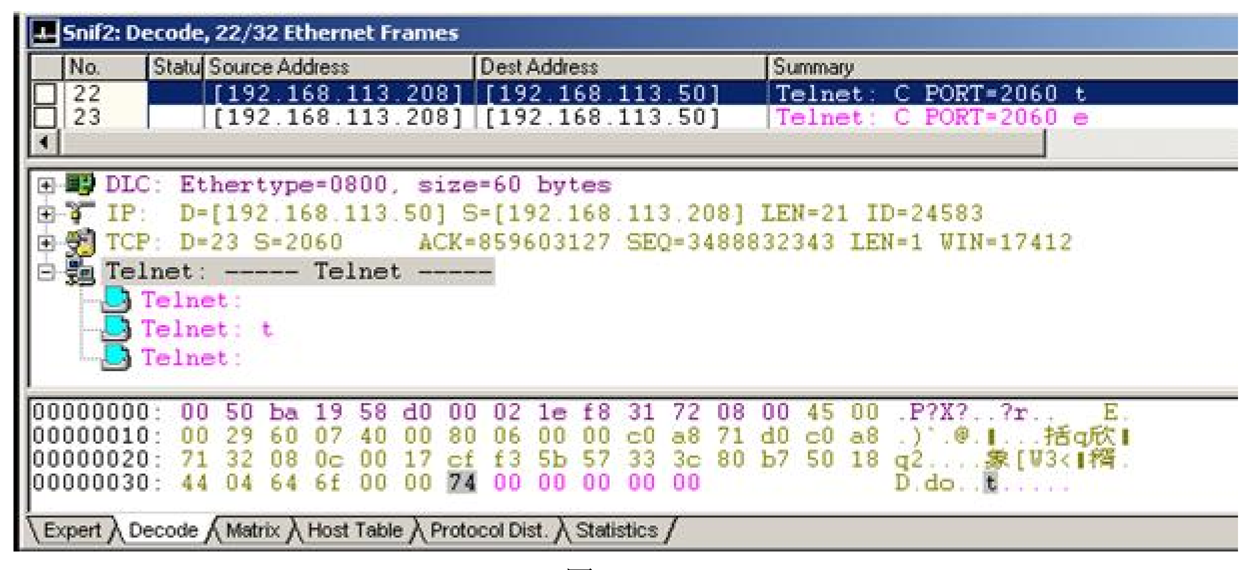


图 18

3、抓 FTP 密码

本例从 192.168.113.208 这台机器 ftp 到 192.168.113.50，用 Sniff Pro 抓到用户名和密码。

步骤 1：设置规则

如图 12 所示，选择 Capture 菜单中的 Defind Filter 出现图 19 界面，选择图 19 中的 ADDress 项，在 station1 和 2 中分别填写两台机器的 IP 地址，选择 Advanced 选项，选择选 IP/TCP/FTP，将 Packet Size 设置为 In Between 63 -71，Packet Type 设置为 Normal。如图 20 所示，选择 Data Pattern 项，点击箭头所指的 Add Pattern 按钮，出现图 21 界面，按图设置 OFFset 为 2F，方格内填入 18，name 可任意起。确定后如图 22 点击 Add NOT 按钮，再点击 Add Pattern 按钮增加第二条规则，按图 23 所示设置好规则，确定后如图 24 所示。

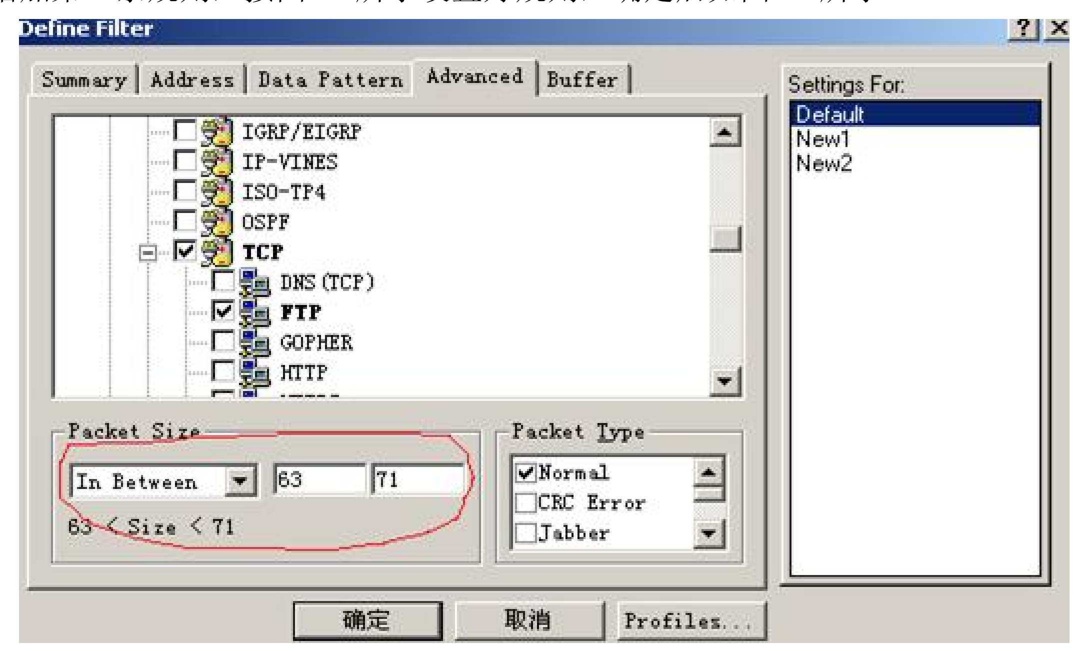


图 19

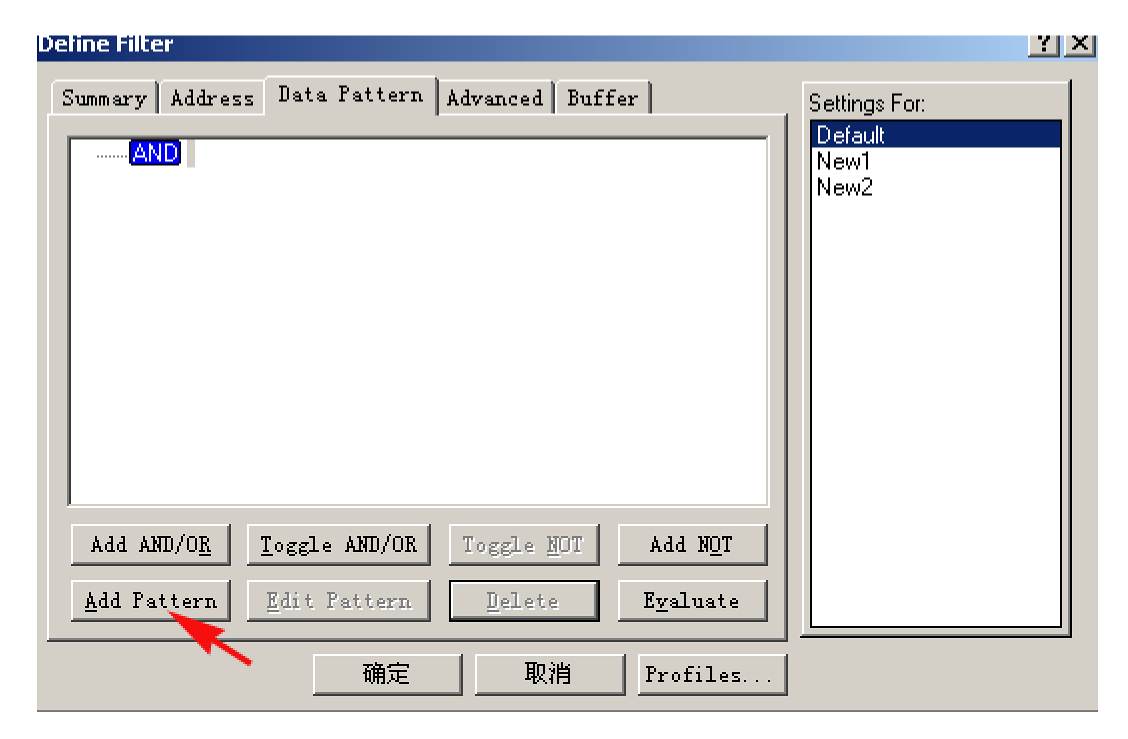

Define Filter
Summary
Address
Data Pattern
Advanced
Buffer
AND
Settings For:
Default
New1
New2
Add AND/OR
Toggle AND/OR
Toggle NOT
Add NOT
Add Pattern
Edit Pattern
Delete
Evaluate
确定
取消
Profiles...

图 20

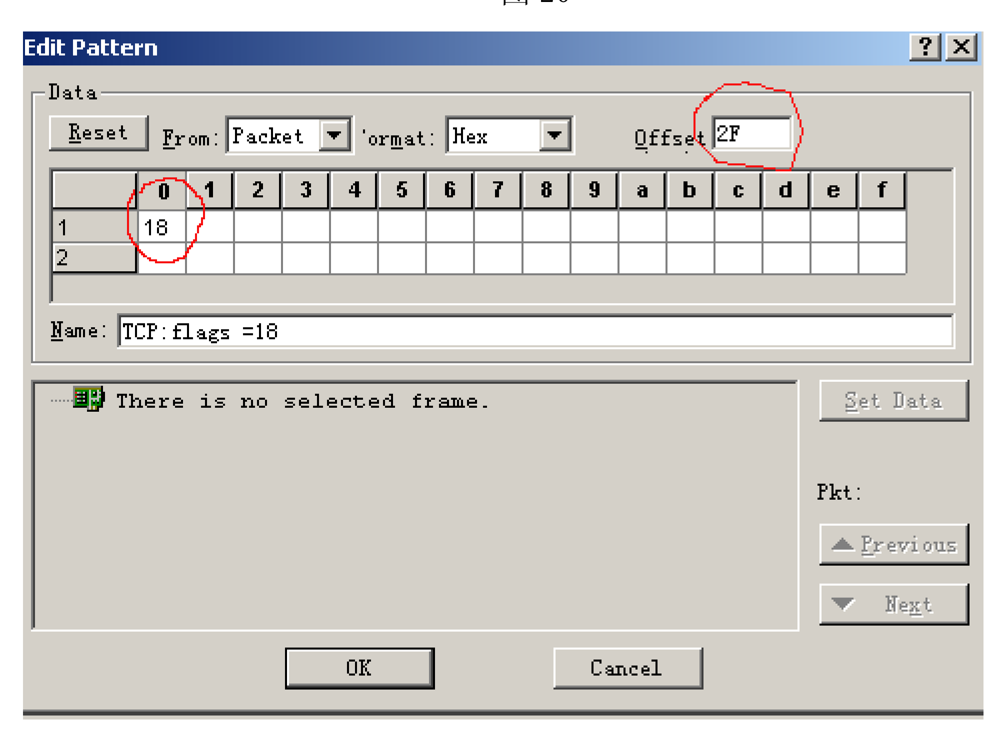

Edit Pattern
Data
Reset
From: Packet
'ormat: Hex
Offset 2F
0 1 2 3 4 5 6 7 8 9 a b c d e f
1 18
2
Name: TCP:flags =18
There is no selected frame.
Set Data
Pkt:
Previous
Next
OK
Cancel

图 21

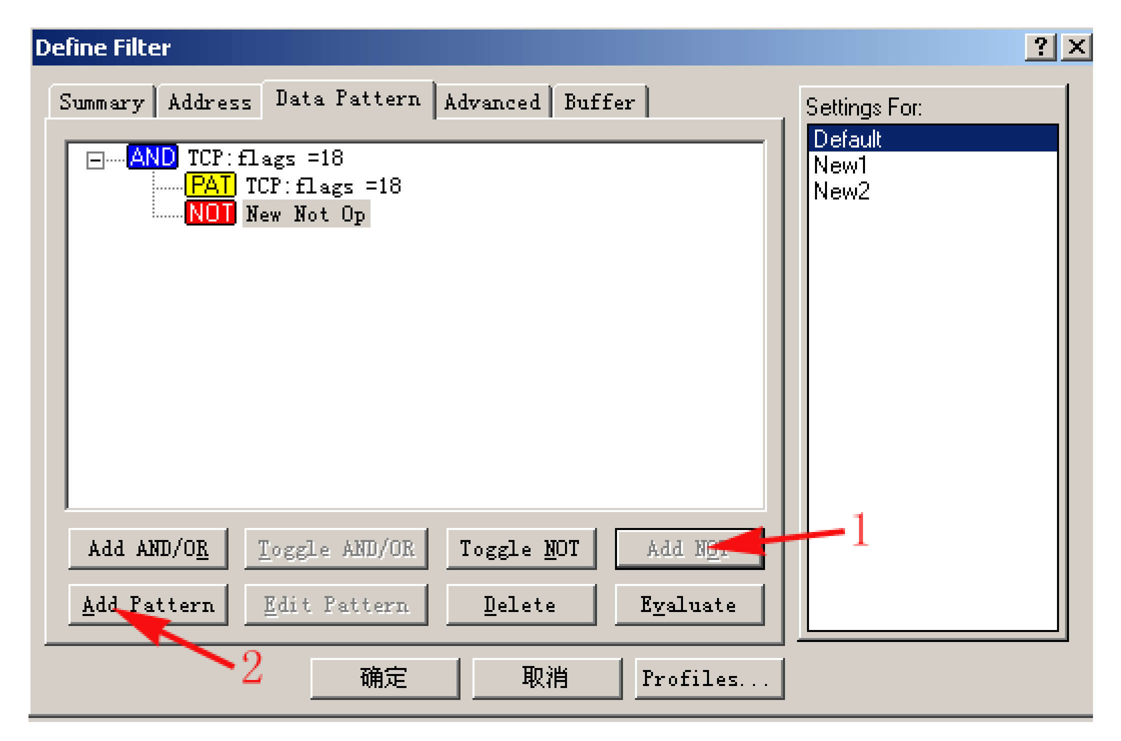


图 22

Edit Pattern

Data

Reset From: Packet Format: Hex Offset: F

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	10															
2																

Name: IP: serviers = 10

There is no selected frame.

Set Data

Pkt:

Previous

Next

OK Cancel

图 23

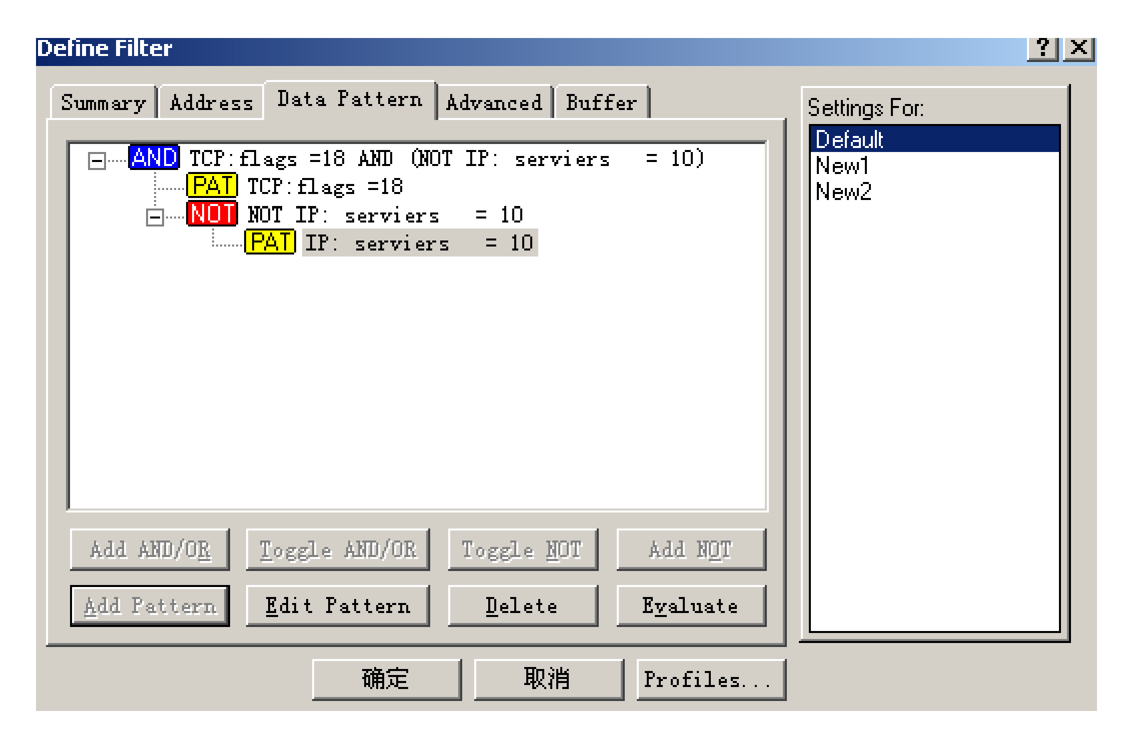


图 24

步骤 2：抓包

按 F10 键出现图 15 界面，开始抓包。

步骤 3：运行 FTP 命令

本例使 FTP 到一台开有 FTP 服务的 Linux 机器上

D:/>ftp 192.168.113.50

Connected to 192.168.113.50.

220 test1 FTP server (Version wu-2.6.1(1) Wed Aug 9 05:54:50 EDT 2000) ready.

User (192.168.113.50:(none)): test

331 Password required for test.

Password:

步骤 4：察看结果

图 16 中箭头所指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 25 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。可以清楚地看出用户名为 test 密码为 123456789。

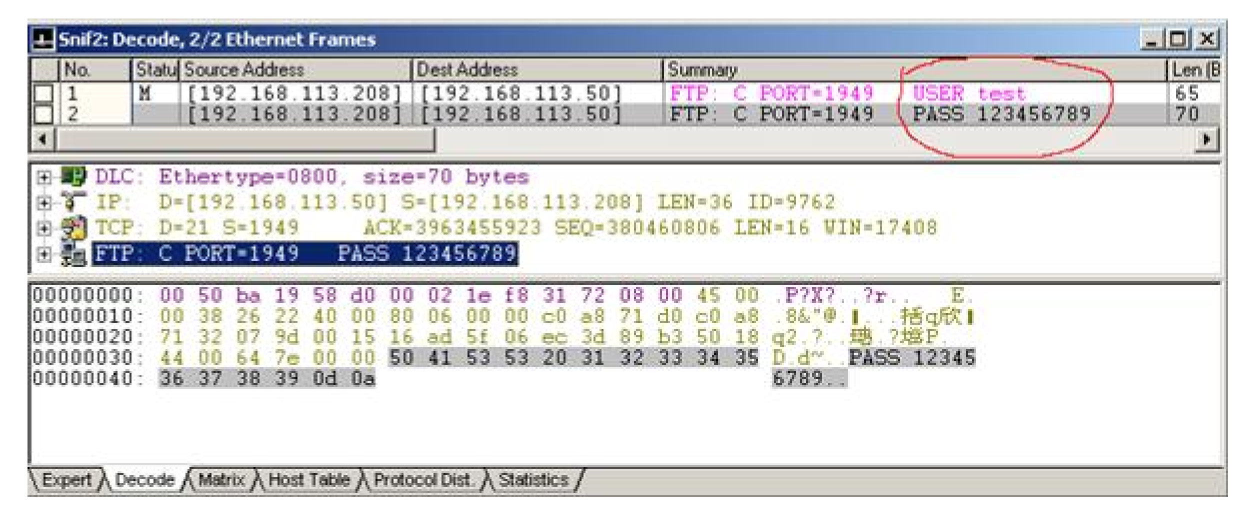


图 25

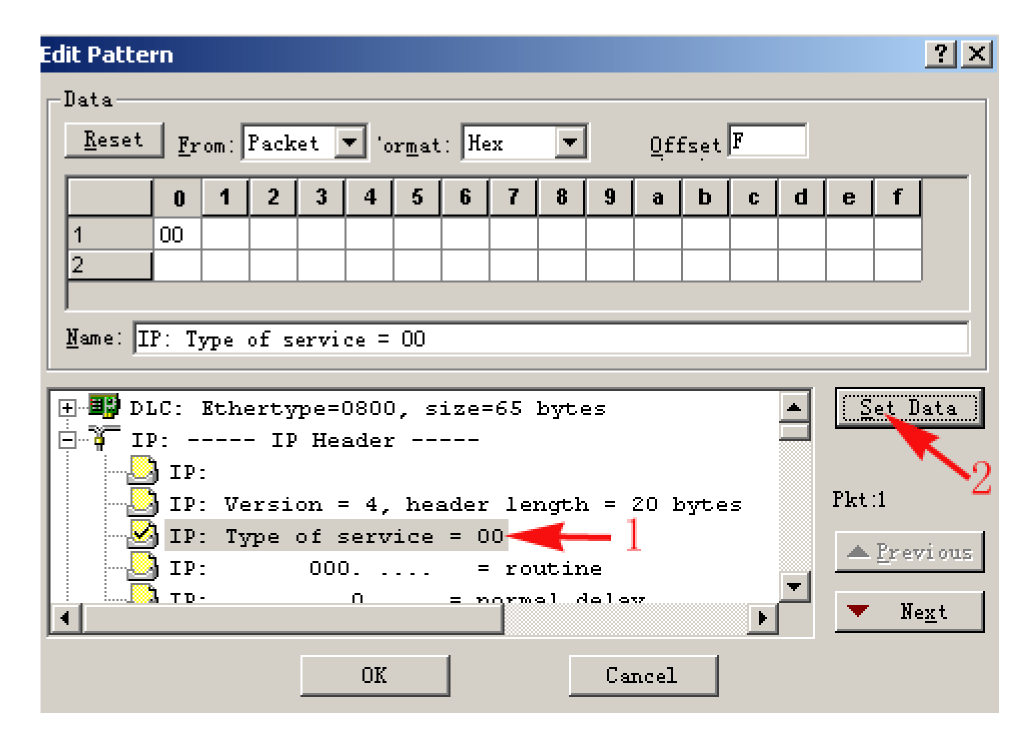
解释：

虽然把密码抓到了，但大家也许设不理解，将图 19 中 Packet Size 设置为 63 -71 是根据用户名和口令的包大小来设置的，图 25 可以看出口令的数据包长度为 70 字节，其中协议头长度为：14+20+20=54，与 telnet 的头长度相同。Ftp 的数据长度为 16，其中关键字 PASS 占 4 个字节，空格占 1 个字节，密码占 9 个字节，0d 0a(回车 换行)占 2 个字节，包长度=54+16=70。如果用户名和密码比较长那么 Packet Size 的值也要相应的增长。

Data Pattern 中的设置是根据用户名和密码中包的特有规则设定的，为了更好的说明这个问题，请在开着图 15 的情况下选择 Capture 菜单中的 Defind Filter，如图 20 所示，选择 Data Pattern 项，点击箭头所指的 Add Pattern 按钮，出现图 26 界面，选择图中 1 所指然后点击 2 所指的 Set Data 按钮。OFFset、方格内、Name 将填上相应的值。

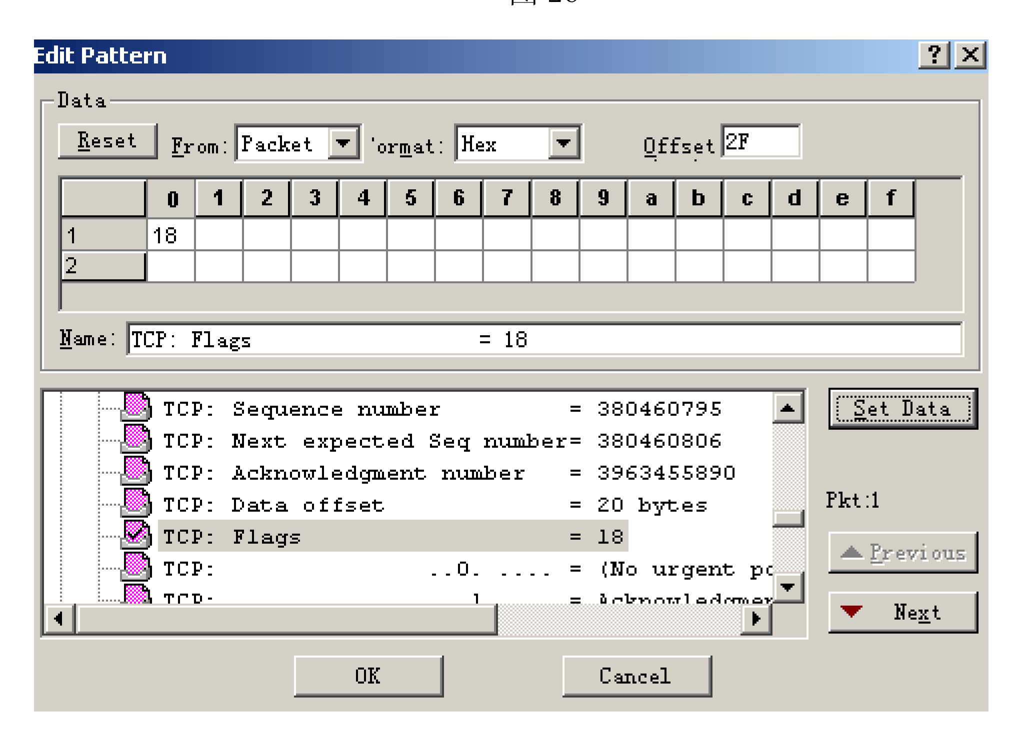
同理图 27 中也是如此。

这些规则的设置都是根据你要抓的包的相应特征来设置的，这些都需要对 TCP/IP 协议的深入了解，从图 28 中可以看出网上传输的都是一位一位的比特流，操作系统将比特流转换为二进制，Sniffer 这类的软件又把二进制换算为 16 进制，然后又为这些数赋予相应的意思，图中的 18 指的是 TCP 协议中的标志位是 18。OFFset 指的是数据包中某位数据的位置，方格内填的是值。



Edit Pattern
Data
Reset
From: Packet
'ormat: Hex
Offset F
0 1 2 3 4 5 6 7 8 9 a b c d e f
1 00
2
Name: IP: Type of service = 00
DLC: Ethertype=0800, size=65 bytes
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: 000. .... = routine
Set Data
Pkt:1
Previous
Next
OK
Cancel
1
2

图 26



Edit Pattern
Data
Reset
From: Packet
'ormat: Hex
Offset 2F
0 1 2 3 4 5 6 7 8 9 a b c d e f
1 18
2
Name: TCP: Flags = 18
TCP: Sequence number = 380460795
TCP: Next expected Seq number= 380460806
TCP: Acknowledgment number = 3963455890
TCP: Data offset = 20 bytes
TCP: Flags = 18
TCP: ..0. .... = (No urgent pc
Set Data
Pkt:1
Previous
Next
OK
Cancel

图 27

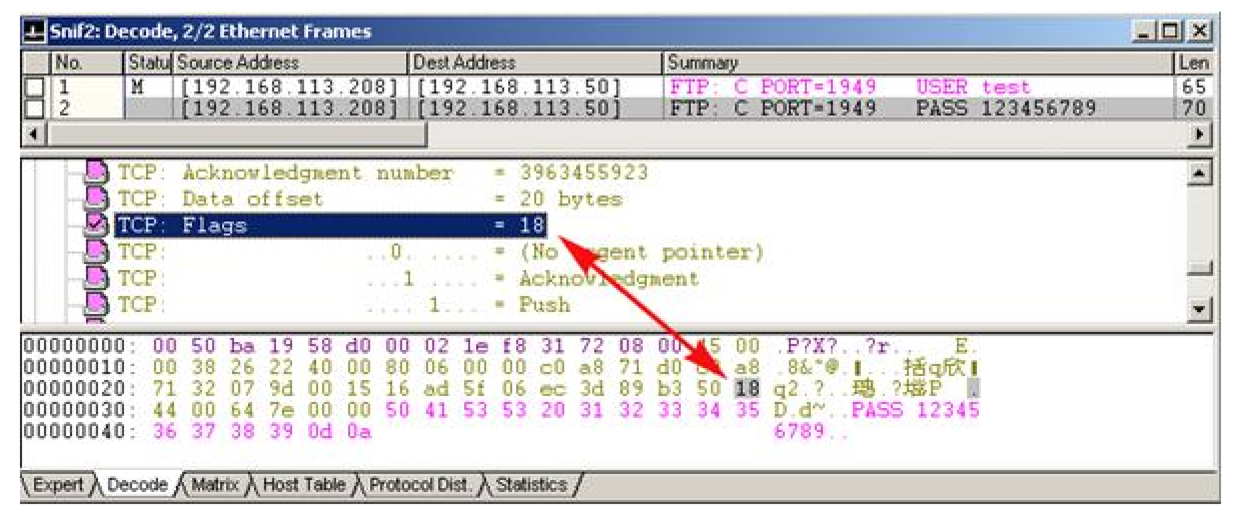


图 28

4、抓 HTTP 密码

步骤 1：设置规则

按照下图 29、30 进行设置规则，设置方法同上。

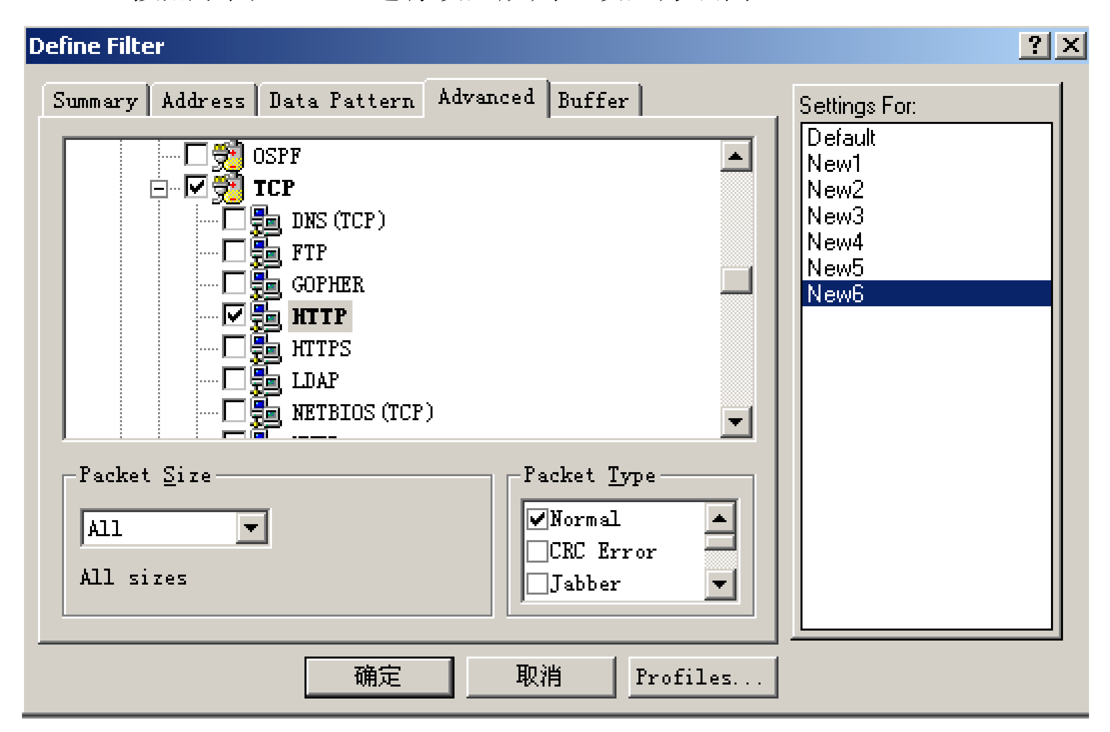


图 29

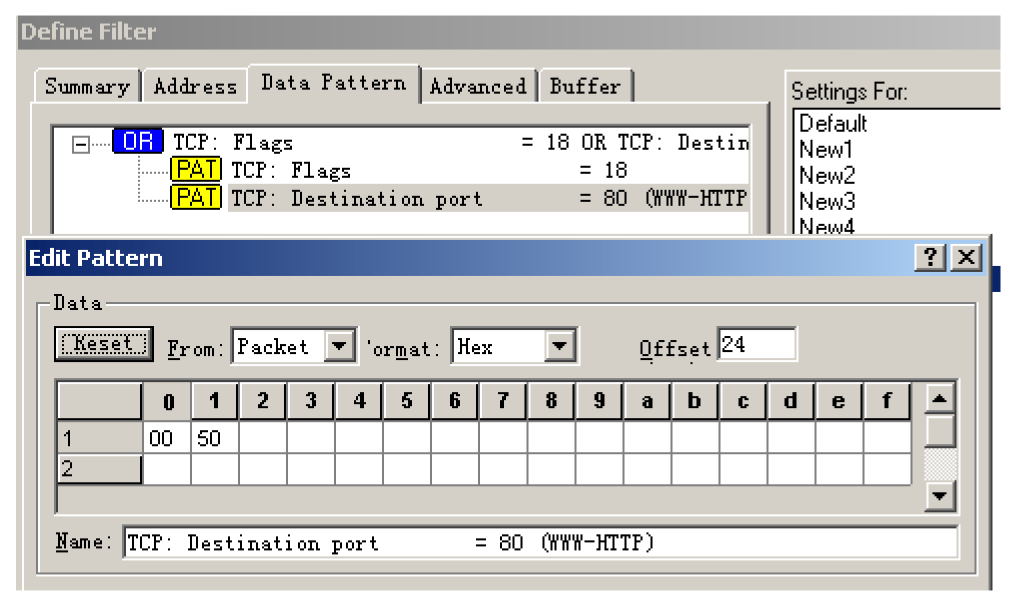


图 30

步骤 2：抓包

按 F10　键开始抓包。

步骤 3：访问www.ccidnet.com网站

步骤 4：察看结果

图 16 中箭头所 指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 31 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。在 Summary 中找到含有 POST 关键字的包，可以清楚地看出用户名为 qiangkn997，密码为?，这可是我邮箱的真实密码！当然不能告诉你，不过欢迎来信进行交流。

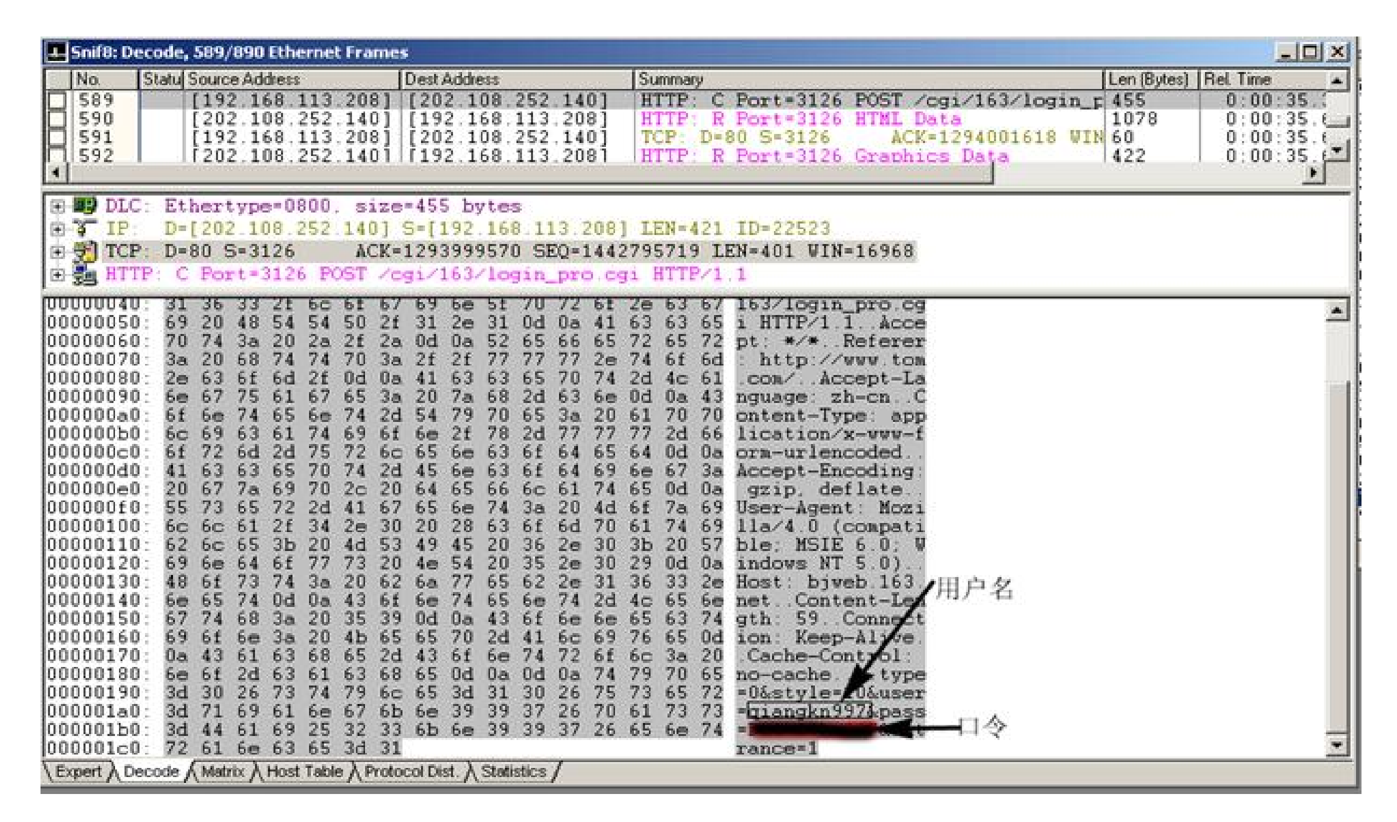


图 31

后记

本文中的例子是网内试验，若捕捉全网机器的有关数据请将图 13 中的 station 设置为 any<->any，作为学习研究可以，可别做坏事！如果要用好 Sniff Pro 必须有扎实的网络基础知识特别是 TCP/IP 协议的知识，其实 Sniff Pro 本身也是学习这些知识的好工具。

用协议分析工具学习 TCP/IP

一、　前言

目前，网络的速度发展非常快，学习网络的人也越来越多，稍有网络常识的人都知道 TCP/IP 协议是网络的基础，是 Internet 的语言，可以说没有 TCP/IP 协议就没有互联网的今天。目前号称搞网的人非常多，许多人就是从一把夹线钳，一个测线器联网开始接触网络的，如果只是联网玩玩，知道几个 Ping 之类的命令就行了，如果想在网络上有更多的发展不管是黑道还是红道，必须要把 TCP/IP 协议搞的非常明白。

学习过 TCP/IP 协议的人多有一种感觉，这东西太抽象了，没有什么数据实例，看完不久就忘了。本文将介绍一种直观的学习方法，利用协议分析工具学习 TCP/IP，在学习的过程中能直观的看到数据的具体传输过程。

为了初学者更容易理解，本文将搭建一个最简单的网络环境，不包含子网。

二、试验环境

1、网络环境

如图 1 所示



图 1

为了表述方便，下文中 2 号机即指地址为 10.1.27.24 的计算机，1 号机指地址为 10.1.27.4 的计算机。

2、操作系统

两台机器都为 Windows 2003 ，1 号机机器作为服务器，安装 FTP 服务

3、协议分析工具

Windows 环境下常用的工具有：Sniffer Pro、Natxray、Iris 以及 windows 2000 自带的网络监视器等。本文选用 Iris 作为协议分析工具。

在客户机 2 号机安装 Sniffer Pro 软件。

三、测试过程

1、测试例子

将 1 号机计算机中的一个文件通过 FTP 下载到 2 号机中。

2、Sniffer Pro 的设置

由于 Sniffer Pro 具有网络监听的功能，如果网络环境中还有其它的机器将抓很多别的数据包，这样为学习带来诸多不便，为了清楚地看清楚上述例子的传输过程首先将 Sniffer Pro 设置为只抓 2 号机和 1 号机之间的数据包。设置过程如下：

1)打开 Sniffer Pro，打开菜单中的“Tools→address book”，填写上两台 PC 的 IP address。如下图：

Name	HW Address	Network Address	Type	Description
10.1.27.4	ETHER	IP 10.1.27.4		
10.1.27.24	ETHER	IP 10.1.27.24		

2）依次打开菜单“capture→Define filter”，找到“address”选项卡，在选项卡中填写如下参数：

（注意：在 Address 中选择 IP，在这个选项卡中还有 IPX，Hardware 选项。然后将“Address book”中的 IP 地址拖动到 station1 与 station2 中）

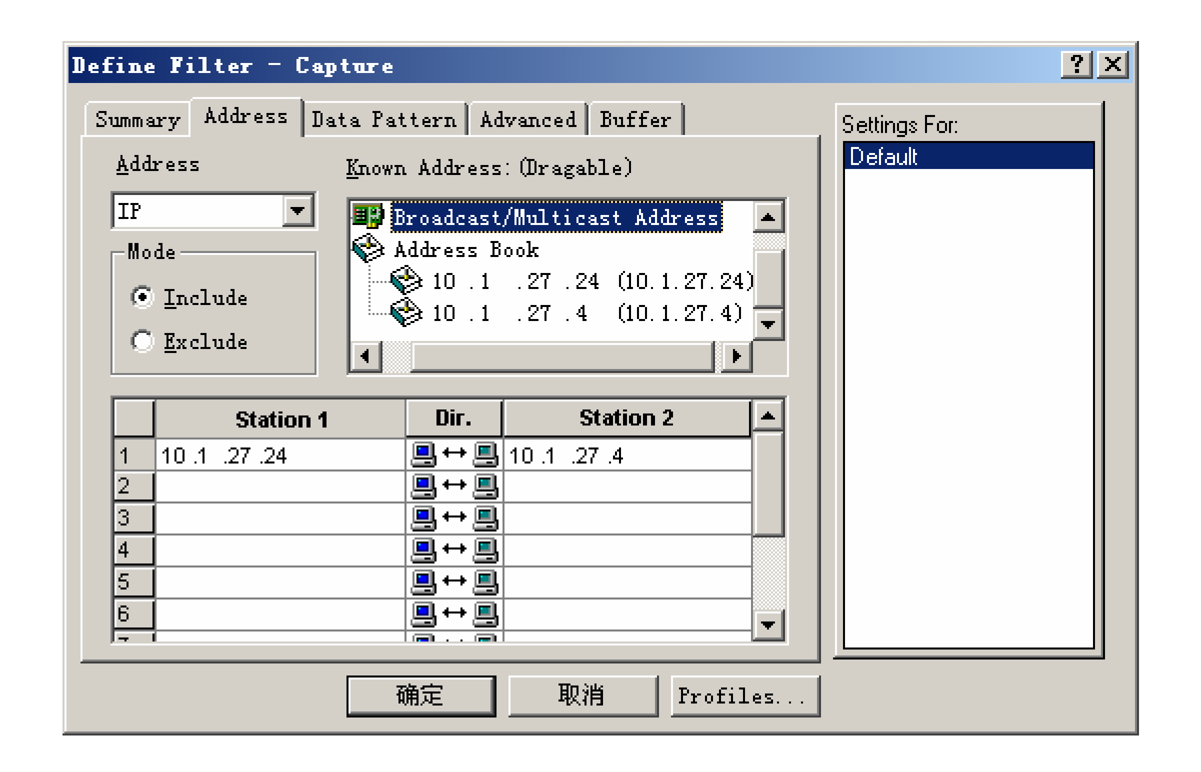


图 3

3、抓包

按下 Sniffer Pro 工具栏中 开始按钮。在浏览器中输入：ftp://10.1.27.4，找到要下载的文件 ，鼠标右键该文件，在弹出的菜单中选择“复制到文件夹”开始下载，下载完后在 Sniffer Pro 工具栏中按 按钮停止抓包。图 4 显示的就是 FTP 的整个过程，下面我们将详细分析这个过程。

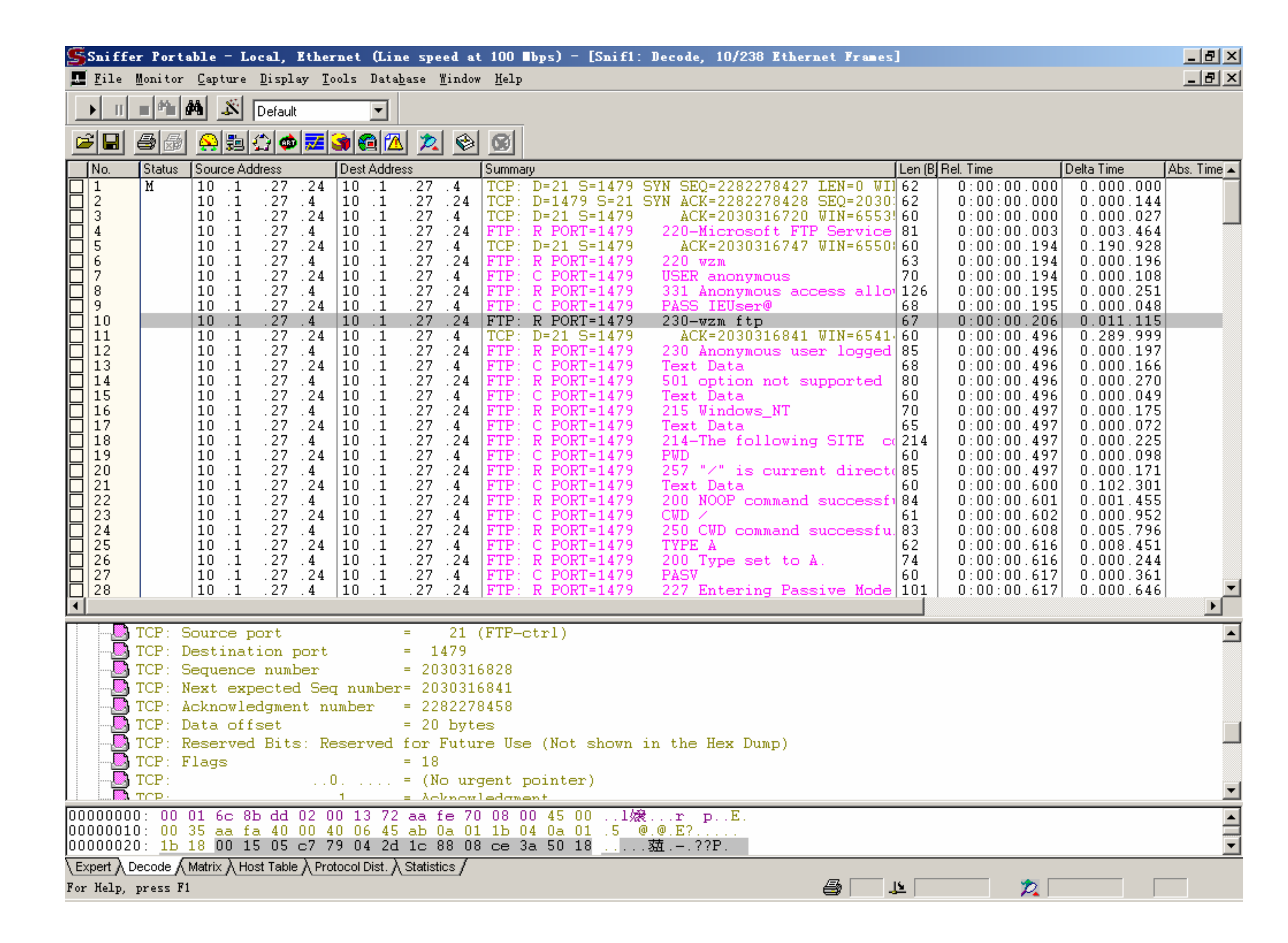


图 4

说明：为了能抓到 ARP 协议的包，在 WINDOWS 2000 中运行 arp－d 清除 arp 缓存。

四、过程分析

1、TCP/IP 的基本原理

本文的重点虽然是根据实例来解析 TCP/IP，但要讲明白下面的过程必须简要讲一下 TCP/IP 的基本原理。

1）网络是分层的，每一层分别负责不同的通信功能。

TCP/IP 通常被认为是一个四层协议系统，TCP/IP 协议族是一组不同的协议组合在一起构成的协议族。尽管通常称该协议族为 TCP/IP，但 TCP 和 IP 只是其中的两种协议而已，如表 1 所示。每一层负责不同的功能：

TCP/IP 层描述	主要协议	主要功能
应用层	HTTP、Telnet、FTP 和 E-mail 等	负责把数据传输到传输层或接收从传输层返回的数据
传输层	TCP 和 UPD	主要为两台主机上的应用程序提供端到端的通信，TCP 为两台主机提供可靠的数据通信。它所做的工作包括把应用程序交给它的数据分成合适的小块交给下面的网络层，确认接收到的分组，设置发送最后确认分组的超时时钟等。UPD 则为应用层提供一种非常简单的服务。它只是把称作数据报的分组从一台主机发送到另一台主机，但并不保证该数据能到达另一端。
网络层	ICMP、IP 和 IGMP	有时称作互联网层，主要为数据包选择路由，其中 IP 是 TCP/IP 协议族中最为核心的协议。所有的 TCP、UPD、ICMP 及 IGMP 数据协议都以 IP 数据包格式传输。
链路层	ARP、RARP 和设备驱动程序及接口卡	发送时将 IP 包作为帧发送；接收时把接收到的位组装成帧；提供链路管理、错误检测等。

表 1

分层的概念说起来非常简单，但在实际的应用中非常的重要，在进行网络设置和排除故障时对网络层次理解得很透，将对工作有很大的帮助。例如：设置路由是网络层 IP 协议的事，要查找 MAC 地址是链路层 ARP 的事，常用的 Ping 命令由 ICMP 协议来做的。

图 5 显示了各层协议的关系，理解它们之间的关系对下面的协议分析非常重要。

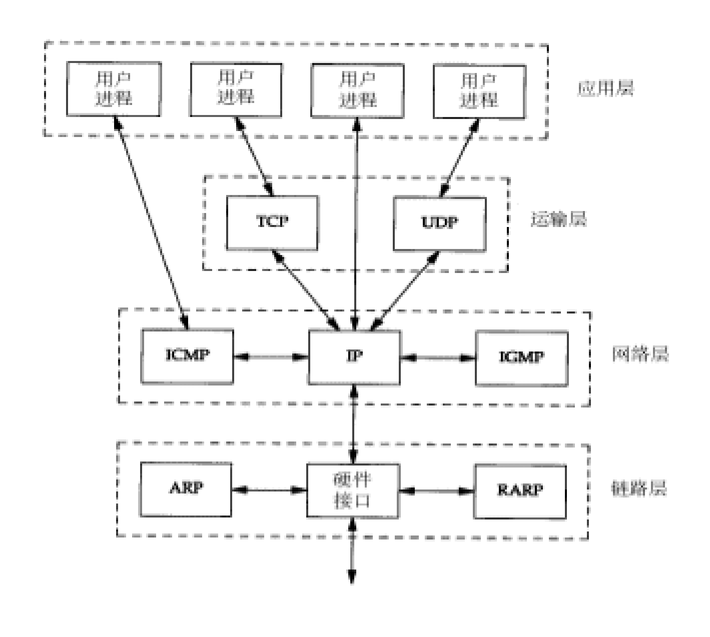


图 5

2）数据发送时是自上而下，层层加码；数据接收时是自下而上，层层解码。

当应用程序用TCP传送数据时，数据被送入协议栈中，然后逐个通过每一层直到被当作一串比特流送入网络。其中每一层对收到的数据都要增加一些首部信息（有时还要增加尾部信息），该过程如图 6 所示。TCP传给IP的数据单元称作TCP报文段或简称为TCP段。I P传给网络接口层的数据单元称作IP数据报。 通过以太网传输的比特流称作帧(Frame)。

数据发送时是按照图 6 自上而下，层层加码；数据接收时是自下而上，层层解码。

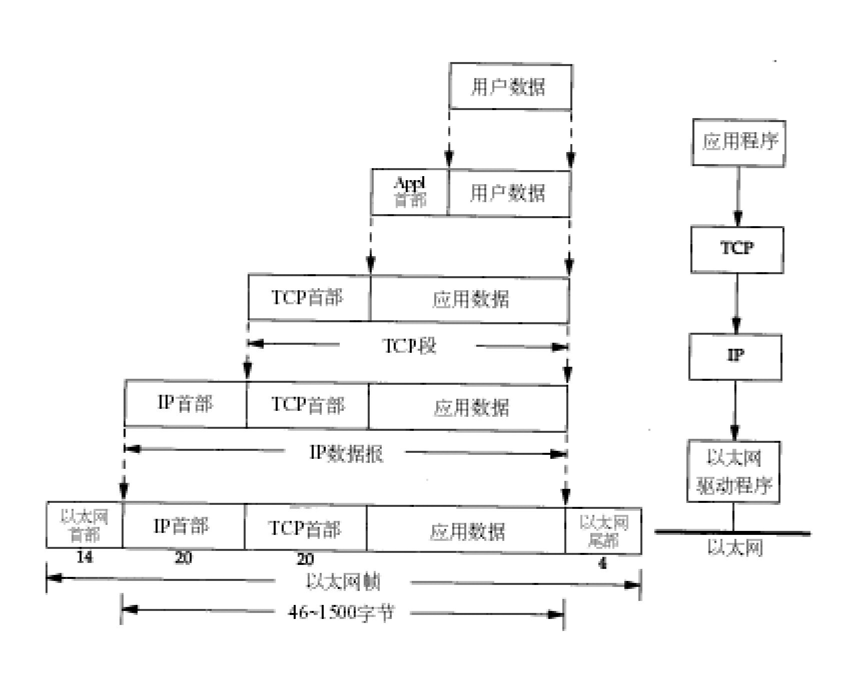


图 6

3）逻辑上通讯是在同级完成的

垂直方向的结构层次是当今普遍认可的数据处理的功能流程。每一层都有与其相邻层的接口。为了通信，两个系统必须在各层之间传递数据、指令、地址等信息，通信的逻辑流程与真正的数据流的不同。虽然通信流程垂直通过各层次，但每一层都在逻辑上能够直接与远程计算机系统的相应层直接通信。

从图 7 可以看出，通讯实际上是按垂直方向进行的，但在逻辑上通信是在同级进行的。

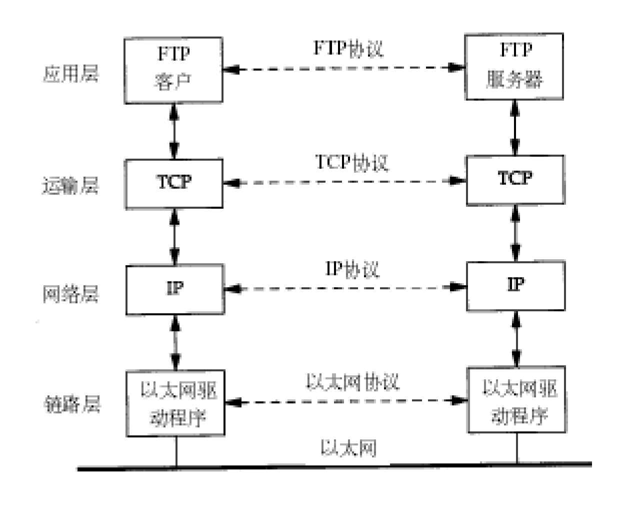


图 7

2、过程描述

为了更好的分析协议，我们先描述一下上述例子数据的传输步骤。如图 8 所示：

1)FTP 客户端请求 TCP 用服务器的 IP 地址建立连接。

2)TCP 发送一个连接请求分段到远端的主机，即用上述 IP 地址发送一份 IP 数据报。

3) 如果目的主机在本地网络上，那么 IP 数据报可以直接送到目的主机上。如果目的主机在一个远程网络上，那么就通过 IP 选路函数来确定位于本地网络上的下一站路由器地址，并让它转发 IP 数据报。在这两种情况下，IP 数据报都是被送到位于本地网络上的一台主机或路由器。

4) 本例是一个以太网，那么发送端主机必须把 32 位的 IP 地址变换成 48 位的以太网地址，该地址也称为 MAC 地址，它是出厂时写到网卡上的世界唯一的硬件地址。把 IP 地址翻译到对应的 MAC 地址是由 ARP 协议完成的。

5) 如图的虚线所示，ARP 发送一份称作 ARP 请求的以太网数据帧给以太网上的每个主机，这个过程称作广播。ARP 请求数据帧中包含目的主机的 IP 地址，其意思是“如果你是这个 IP 地址的拥有者，请回答你的硬件地址。”

6) 目的主机的 ARP 层收到这份广播后，识别出这是发送端在寻问它的 IP 地址，于是发送一个 ARP 应答。这个 ARP 应答包含 I P 地址及对应的硬件地址。

7）收到 ARP 应答后，使 ARP 进行请求—应答交换的 IP 数据包现在就可以传送了。

8）发送 IP 数据报到目的主机。

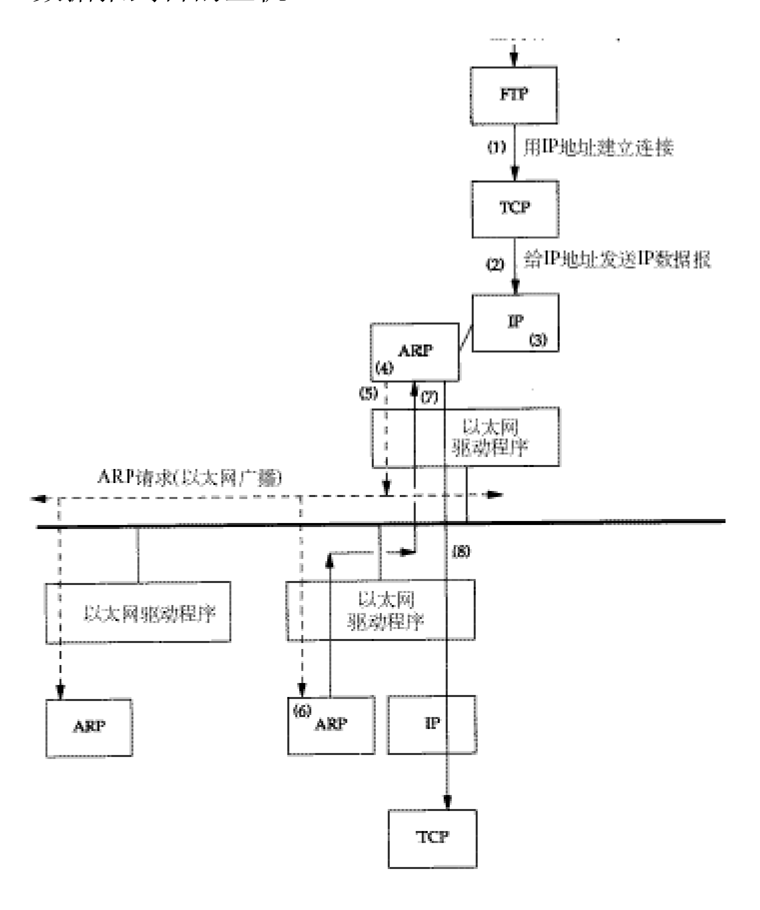


图 8

3、实例分析

下面通过分析用 Sniffer 捕获的包来分析一下 TCP/IP 的工作过程，为了更清晰的解释数据传送的过程，我们按传输的不同阶段抓了四组数据，分别是查找服务器、建立连接、数据传输和终止连接。每组数据，按下面三步进行解释。

显示数据包

解释该数据包

按层分析该包的头信息

第一组 查找服务器

1）下图显示的是 1、2 行的数据

Name	HW Address	Network Address	Type	Description
10.1 .27.4	ETHER	IP 10.1.27.4		
10.1 .27.24	ETHER	IP 10.1.27.24		

图 9

2）解释数据包

这两行数据就是查找服务器及服务器应答的过程。

在第 1 行中，源端主机的 IP 地址是 10.1.27.24。目的端主机的 IP 地址是 FF:FF:FF:FF:FF:FF，这个地址是十六进制表示的，F 换算为二进制就是 1111，全 1 的地址就是广播地址。所谓广播就是向本网上的每台网络设备发送信息，电缆上的每个以太网接口都要接收这个数据帧并对它进行处理，这一行反映的是步骤 5）的内容，ARP 发送一份称作 ARP 请求的以太网数据帧给以太网上的每个主机。网内的每个网卡都接到这样的信息“谁是 10.1.27.4 的 IP 地址的拥有者，请将你的硬件地址告诉我”。

第 2 行反映的是步骤 6）的内容。在同一个以太网中的每台机器都会"接收"到这个报文，但正常状态下除了 1 号机外其他主机应该会忽略这个报文，而 1 号的主机的 ARP 层收到这份广播报文后，识别出这是发送端在寻问它的 IP 地址，于是发送一个 ARP 应答。告知自己的 IP 地址和 MAC 地址。第 2 行可以清楚的看出 1 号回答的信息__自己的 MAC 地址 00:50:FC:22:C7:BE。

这两行反映的是数据链路层之间一问一答的通信过程。这个过程就像我要在一个坐满人的教室找一个叫“张三”的人，在门口喊了一声“张三”，这一声大家都听见了，这就叫广播。张三听到后做了回应，别人听到了没做回应，这样就与张三取得了联系。

3）头信息分析

如下图左栏所示，第 1 数据包包含了两个头信息：以太网（Ethernet）和 ARP。

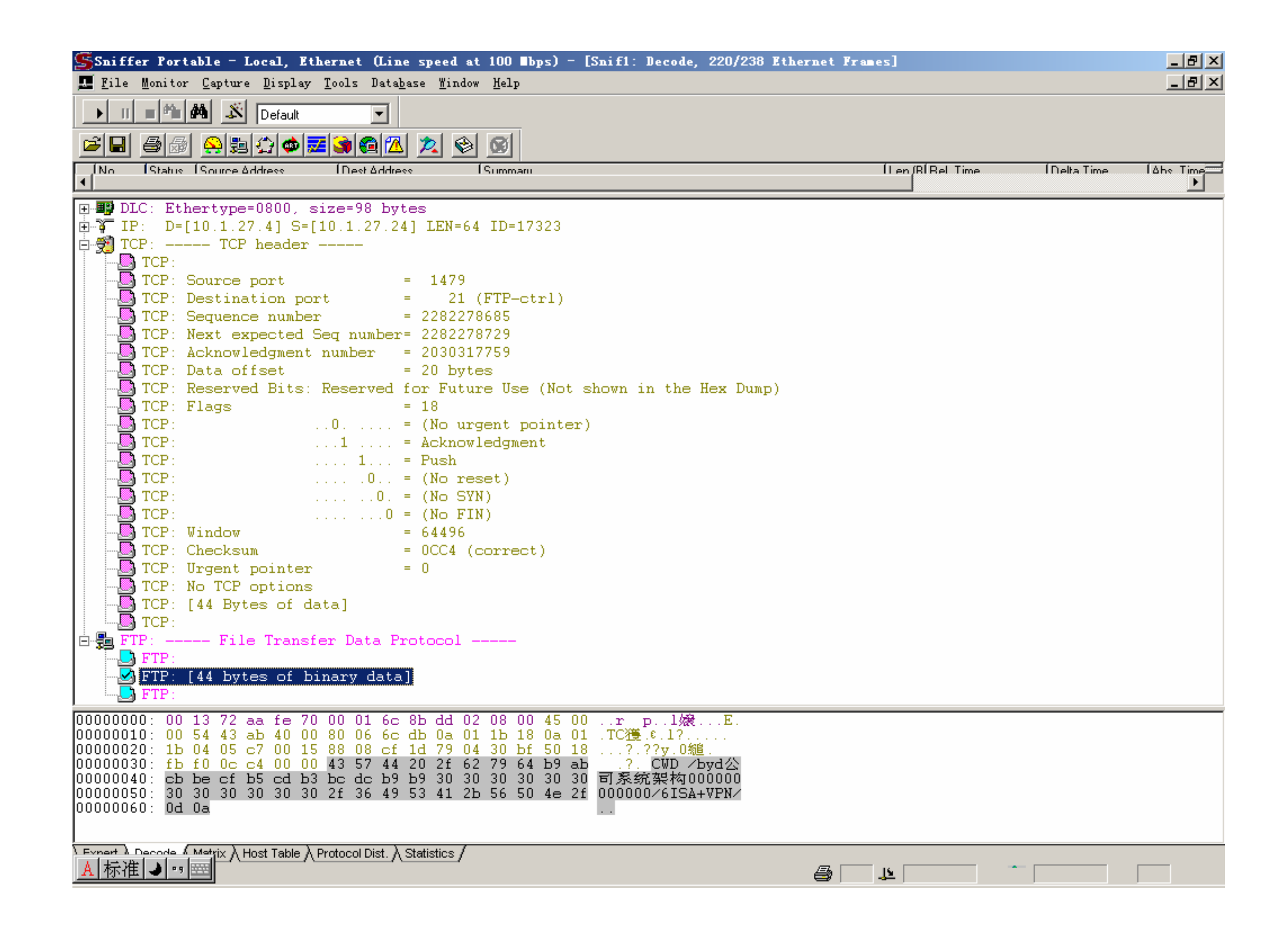


图 10

No.	Status	Source Address	Dest Address	Summary	Len (By	Rel. Time	Delta Time	Abs. Tim
1	M	10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1479 SYN SEQ=2282278427 LEN=0 WI	62	0:00:00.000	0.000.000	
2		10 .1 .27 .4	10 .1 .27 .24	TCP: D=1479 S=21 SYN ACK=2282278428 SEQ=2030	62	0:00:00.000	0.000.144	
3		10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1479 ACK=2030316720 WIN=6553	60	0:00:00.000	0.000.027	
4		10 .1 .27 .4	10 .1 .27 .24	FTP: R PORT=1479 220-Microsoft FTP Service	81	0:00:00.003	0.003.464	
5		10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1479 ACK=2030316747 WIN=6550	60	0:00:00.194	0.190.928	

下表 2 是以太网的头信息，括号内的数均为该字段所占字节数，以太网报头中的前两个字段是以太网的源地址和目的地址。目的地址为全 1 的特殊地址是广播地址。电缆上的所有以太网接口都要接收广播的数据帧。两个字节长的以太网帧类型表示后面数据的类型。对于 ARP 请求或应答来说，该字段的值为 0806。

第 2 行中可以看到，尽管 ARP 请求是广播的，但是 ARP 应答的目的地址却是 1 号机的（00 50 FC 22 C7 BE）。ARP 应答是直接送到请求端主机的。

行	以太网目的地址（6）	以太网源地址（6）	帧类型（2）
1	FF FF FF FF FF FF	00 50 FC 22 C7 BE	06 06
2	00 50 FC 22 C7 BE	00 50 27 F6 50 53	06 06

表 2

下表 3 是 ARP 协议的头信息。硬件类型字段表示硬件地址的类型。它的值为 1 即表示以太网地址。协议类型字段表示要映射的协议地址类型。它的值为 0800 即表示 IP 地址。它的值与包含 I P 数据报的以太网数据帧中的类型字段的值相同。接下来的两个 1 字节的字段，硬件地址长度和协议地址长度分别指出硬件地址和协议地址的长度，以字节为单位。对于以太网上 IP 地址的 ARP 请求或应答来说，它们的值分别为 6 和 4。Op 即操作（Opoperation），1 是 ARP 请求、2 是 ARP 应答、3 是 RARP 请求和 4 为 RARP 应答，第二行中该字段值为 2 表示应答。接下来的四个字段是发送端的硬件地址、发送端的 IP 地址、目的端的硬件地址和目的端 IP 地址。注意，这里有一些重复信息：在以太网的数据帧报头中和 ARP 请求数据帧中都有发送端的硬件地址。对于一个 ARP 请求来说，除目的端硬件地址外的所有其他的字段都有填充值。

表 3 的第 2 行为应答，当系统收到一份目的端为本机的 ARP 请求报文后，它就把硬件地址填进去，然后用两个目的端地址分别替换两个发送端地址，并把操作字段置为 2，最后把它发送回去。

行	硬件类型（2）	协议类型（2）	硬件地址长度（1）	协议地址长度（1）	Op（2）	发送端以太网地址（6）	发送端 IP 地址（4）	目的以太网地址（6）	目的 IP 地址（4）
1	00 00	08 00	06	06	00 01	0050FC22C7BE	C0A871D0	000000000000	C0A871D1
2	00 00	08 00	06	06	00 02	005027F65053	C0A871D1	0050FC22C7BE	C0A871D0

表 3

第二组 建立连接

1）下图显示的是 3-5 行的数据

No.	Status	Source Address	Dest Address	Summary	Len (By	Rel. Time	Delta Time	Abs. Tim
1	M	10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1479 SYN SEQ=2282278427 LEN=0 WI	62	0:00:00.000	0.000.000	
2		10 .1 .27 .4	10 .1 .27 .24	TCP: D=1479 S=21 SYN ACK=2282278428 SEQ=2030	62	0:00:00.000	0.000.144	
3		10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1479 ACK=2030316720 WIN=6553	60	0:00:00.000	0.000.027	

图 11

2）解释数据包

这三行数据是两机建立连接的过程。

这三行的核心意思就是 TCP 协议的三次握手。TCP 的数据包是靠 IP 协议来

传输的。但 IP 协议是只管把数据送到出去，但不能保证 IP 数据报能成功地到达目的地，保证数据的可靠传输是靠 TCP 协议来完成的。当接收端收到来自发送端的信息时，接受端详发送短发送一条应答信息，意思是：“我已收到你的信息了。”第三组数据将能看到这个过程。TCP 是一个面向连接的协议。无论哪一方向另一方发送数据之前，都必须先在双方之间建立一条连接。建立连接的过程就是三次握手的过程。

这个过程就像要我找到了张三向他借几本书，第一步：我说：“你好，我是担子”，第二步：张三说：“你好，我是张三”，第三步：我说：“我找你借几本书。”这样通过问答就确认对方身份，建立了联系。

下面来分析一下此例的三次握手过程。

1))请求端 208 号机发送一个初始序号（SEQ）987694419 给 1 号机。

2))服务器 1 号机收到这个序号后，将此序号加 1 值为 987694419 作为应答信号（ACK），同时随机产生一个初始序号（SEQ）1773195208，这两个信号同时发回到请求端 208 号机，意思为：“消息已收到，让我们的数据流以 1773195208 这个数开始。”

3))请求端 208 号机收到后将确认序号设置为服务器的初始序号（SEQ）1773195208 加 1 为 1773195209 作为应答信号。

以上三步完成了三次握手，双方建立了一条通道，接下来就可以进行数据传输了。

下面分析 TCP 头信息就可以看出，在握手过程中 TCP 头部的相关字段也发生了变化。

3）头信息分析

如图 12 所示，第 3 数据包包含了三头信息：以太网（Ethernet）和 IP 和 TCP。

头信息少了 ARP 多了 IP、TCP，下面的过程也没有 ARP 的参与，可以这样理解，在局域网内，ARP 负责的是在众多联网的计算机中找到需要找的计算机，找到工作就完成了。

以太网的头信息与第 1、2 行不同的是帧类型为 0800，指明该帧类型为 IP。

```
IP: ----- IP Header -----
  IP:
  IP: Version = 4, header length = 20 bytes
  IP: Type of service = 00
  IP:       000. ....  = routine
  IP:       ...0 .... = normal delay
  IP:       .... 0... = normal throughput
  IP:       .... .0.. = normal reliability
  IP:       .... ..0. = ECT bit - transport protocol will ignore the CE bit
  IP:       .... ...0 = CE bit - no congestion
  IP: Total length    = 45 bytes
  IP: Identification  = 17164
  IP: Flags           = 4X
  IP:       .1.. .... = don't fragment
  IP:       ..0. .... = last fragment
  IP: Fragment offset = 0 bytes
  IP: Time to live    = 128 seconds/hops
  IP: Protocol        = 6 (TCP)
  IP: Header checksum = 6DA1 (correct)
  IP: Source address      = [10.1.27.24], 10 .1  .27 .24
  IP: Destination address = [10.1.27.4], 10 .1  .27 .4
  IP: No options
  IP:
TCP: D=21 S=1479     ACK=2030317074 SEQ=2282278489 LEN=5 WIN=65181
FTP: ----- File Transfer Data Protocol -----
  FTP:

00000000: 00 13 72 aa fe 70 00 01 6c 8b dd 02 08 00 45 00  ..r  p..l獗...E.
00000010: 00 2d 43 0c 40 00 80 06 6d a1 0a 01 1b 18 0a 01  .-C.@.▌.m?.....
00000020: 1b 04 05 c7 00 15 88 08 ce 59 79 04 2e 12 50 18  ...?.?婴y...P.
00000030: fe 9d c5 52 00 00 50 57 44 0d 0a 00             齘臨..PWD...
```

图 12

IP 协议头信息

IP 是 TCP/IP 协议族中最为核心的协议。从图 5 可以看出所有的 TCP、UDP、ICMP 及 IGMP 数据都以 IP 数据报格式传输的，有个形象的比喻 IP 协议就像运货的卡车，将一车车的货物运向目的地。主要的货物就是 TCP 或 UDP 分配给它的。需要特别指出的是 IP 提供不可靠、无连接的数据报传送，也就是说 I P 仅提供最好的传输服务但不保证 IP 数据报能成功地到达目的地。看到这你会不会担心你的 E_MAIL 会不会送到朋友那，其实不用担心，上文提过保证数据正确到达目的地是 TCP 的工作，稍后我们将详细解释。

如表 4 是 IP 协议的头信息。

<table>
<tr><td colspan="5">32位</td><td></td></tr>
<tr><td>4位版本</td><td>4位首部长度</td><td>8位服务类型(TOS)</td><td colspan="2">16位总长度(字节数)</td><td rowspan="5">20字节</td></tr>
<tr><td colspan="3">16位标识</td><td>3位标志</td><td>13位片偏移</td></tr>
<tr><td colspan="2">8位生存时间(TTL)</td><td>8位协议</td><td colspan="2">16位首部检验和</td></tr>
<tr><td colspan="5">32位源IP地址</td></tr>
<tr><td colspan="5">32位目的IP地址</td></tr>
<tr><td colspan="5">选项(如果有)</td><td></td></tr>
<tr><td colspan="5">数据</td><td></td></tr>
</table>

表 4 IP 数据报格式及首部中的各字段

图 12 中所宣布分 45 00—71 01 为 IP 的头信息。这些数是十六进制表示的。一个数占 4 位，例如：4 的二进制是 0100

4 位版本：

表示目前的协议版本号，数值是 4 表示版本为 4，因此 IP 有时也称作 IPv4；

4 位首部长度：

头部的是长度，它的单位是 32 位(4 个字节)，数值为 5 表示 IP 头部长度为 20 字节。

8 位服务类型(TOS)：

00，这个 8 位字段由 3 位的优先权子字段，现在已经被忽略，4 位的 TOS 子字段以及 1 位的未用字段（现在为 0）构成。4 位的 TOS 子字段包含：最小延时、最大吞吐量、最高可靠性以及最小费用构成，这四个 1 位最多只能有一个为 1，本例中都为 0，表示是一般服务。

16 位总长度：

总长度字段是指整个 IP 数据报的长度，以字节为单位。数值为 00 30，换算为十进制为 48 字节，48 字节=20 字节 的 IP 头+28 字节的 TCP 头，这个数据报只是传送的控制信息，还没有传送真正的数据，所以目前看到的总长度就是报头的长度。

16 位标识：

标识字段唯一地标识主机发送的每一份数据报。通常每发送一份报文它的值就会加 1，第 3 行为数值为 30 21，第 5 行为 30 22，第 7 行为 30 23。分片时涉及到标志字段和片偏移字段，本文不讨论这两个字段。

8 位生存时间（TTL）：

TTL（time-to-live）生存时间字段设置了数据报可以经过的最多路由器数。它指定了数据报的生存时间。ttl 的初始值由源主机设置，一旦经过一个处理它的路由器，它的值就减去 1。可根据 TTL 值判断服务器是什么系统和经过的路由器。本例为 80，换算成十进制为 128，WINDOWS 操作系统 TTL 初始值一般为 128，UNIX 操作系统初始值为 255，本例表示两个机器在同一网段且操作系统为 WINDOWS。

8 位协议：

表示协议类型，6 表示传输层是 TCP 协议。

16 位首部检验和：

当收到一份 IP 数据报后，同样对首部中每个 16 位进行二进制反码的求和。由于接收方在计算过程中包含了发送方存在首部中的检验和，因此，如果首部在传输过程中没有发生任何差错，那么接收方计算的结果应该为全 1。如果结果不是全 1，即检验和错误，那么 IP 就丢弃收到的数据报。但是不生成差错报文，由上层去发现丢失的数据报并进行重传。

32 位源 IP 地址和 32 位目的 IP 地址：

实际这是 IP 协议中核心的部分，但介绍这方面的文章非常多，本文搭建的又是一个最简单的网络结构，不涉及路由，本文对此只做简单介绍，相关知识请参阅其它文章。32 位的 IP 地址由一个网络 ID 和一个主机 ID 组成。本例源 IP 地址为 C0 A8 71 D0，转换为十进制为：192.168.113.208；目的 IP 地址为 C0 A8 71 01，转换为十进制为：192.168.113.1。网络地址为 192.168.113，主机地址分别为 1 和 208，它们的网络地址是相同的所以在一个网段内，这样数据在传送过程中可直接到达。

TCP 协议头信息

如表 5 是 ICP 协议的头信息。

<table>
<tr><td colspan="9">32位</td><td></td></tr>
<tr><td colspan="8">16位源端口号</td><td>16位目的端口号</td><td rowspan="5">20字节</td></tr>
<tr><td colspan="9">32位序号</td></tr>
<tr><td colspan="9">32位确认序号</td></tr>
<tr><td>4位首部长度</td><td>保留(6位)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16位窗口大小</td></tr>
<tr><td colspan="8">16位检验和</td><td>16位紧急指针</td></tr>
<tr><td colspan="9">选项</td><td></td></tr>
<tr><td colspan="9">数据</td><td></td></tr>
</table>

表 5 TCP 包首部

第三行 TCP 的头信息是：04 28 00 15 3A DF 05 53 00 00 00 00 70 02 40 00 9A 8D 00 00 02 04 05 B4 01 01 04 02

端口号：

常说 FTP 占 21 端口、HTTP 占 80 端口、TELNET 占 23 端口等，这里指的端口就是 TCP 或 UDP 的端口，端口就像通道两端的门一样，当两机进行通讯时门必须是打开的。源端口和目的端口各占 16 位，2 的 16 次方等于 65536，这就是每台电脑与其它电脑联系所能开的“门”。一般作为服务一方每项服务的端口号是固定的。本例目的端口号为 00 15，换算成十进制为 21，这正是 FTP 的默认端口，需要指出的是这是 FTP 的控制端口，数据传送时用另一端口，第三组的分析能看到这一点。客户端与服务器联系时随机开一个大于 1024 的端口，本例为 04 28，换算成十进制为 1064。你的电脑中了木马也会开一个服务端口。观察端口非常重要，不但能看出本机提供的正常服务，还能看出不正常的连接。Windows 察看端口的命令时 netstat。

32 位序号：

也称为顺序号（Sequence Number），简写为 SEQ，从上面三次握手的分析可以看出，当一方要与另一方联系时就发送一个初始序号给对方，意思是：“让我们建立联系吧？”，服务方收到后要发个独立的序号给发送方，意思是“消息收到，数据流将以这个数开始。”由此可看出，TCP 连接完全是双向的，即双方的数据流可同时传输。在传输过程中双方数据是独立的，因此每个 TCP 连接必须有两个顺序号分别对应不同方向的数据流。

32 位确认序号：

也称为应答号（Acknowledgment Number），简写为 ACK。在握手阶段，确认序号将发送方的序号加 1 作为回答，在数据传输阶段，确认序号将发送方的序号加发送的数据大小作为回答，表示确实收到这些数据。在第三组的分析中将看到这一过程。

4 位首部长度：

这个字段占 4 位，它的单位时 32 位（4 个字节）。本例值为 7，TCP 的头长度为 28 字节，等于正常的长度 2 0 字节加上可选项 8 个字节。，TCP 的头长度最长可为 60 字节（二进制 1111 换算为十进制为 15，15*4 字节=60 字节）。

6 个标志位：

URG 紧急指针，告诉接收 TCP 模块紧要指针域指着紧要数据

ACK 置 1 时表示确认号（为合法，为 0 的时候表示数据段不包含确认信息，确认号被忽略。

PSH 置 1 时请求的数据段在接收方得到后就可直接送到应用程序，而不必等到缓冲区满时才传送。

RST 置 1 时重建连接。如果接收到 RST 位时候，通常发生了某些错误。

SYN 置 1 时用来发起一个连接。

FIN 置 1 时表示发端完成发送任务。用来释放连接，表明发送方已经没有数据发送了。

图 13 的 3 个图分别为 3-5 行 TCP 协议的头信息，这三行是三次握手的过程，我们看看握手的过程标志位发生了什么？

如图 13-1 图请求端 2 号机发送一个初始序号（SEQ）2282278427 给 1 号机。标志位 SYN 置为 1。

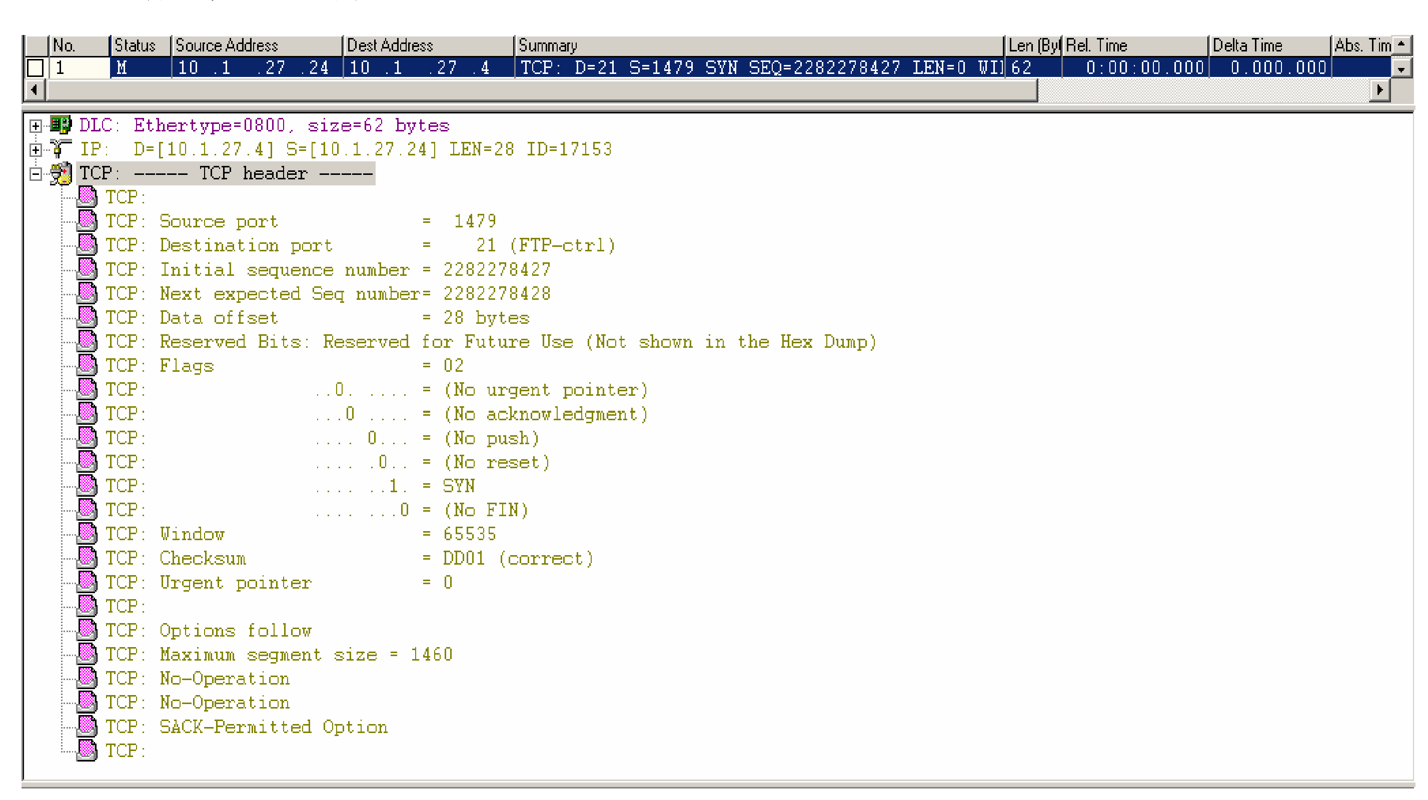


图 13-1

如图 13-2 服务器 1 号机收到这个序号后，将应答信号（ACK）和随机产生一个初始序号（SEQ）2030316719 发回到请求端 2 号机，因为有应答信号和初始序号，所以标志位 ACK 和 SYN 都置为 1。

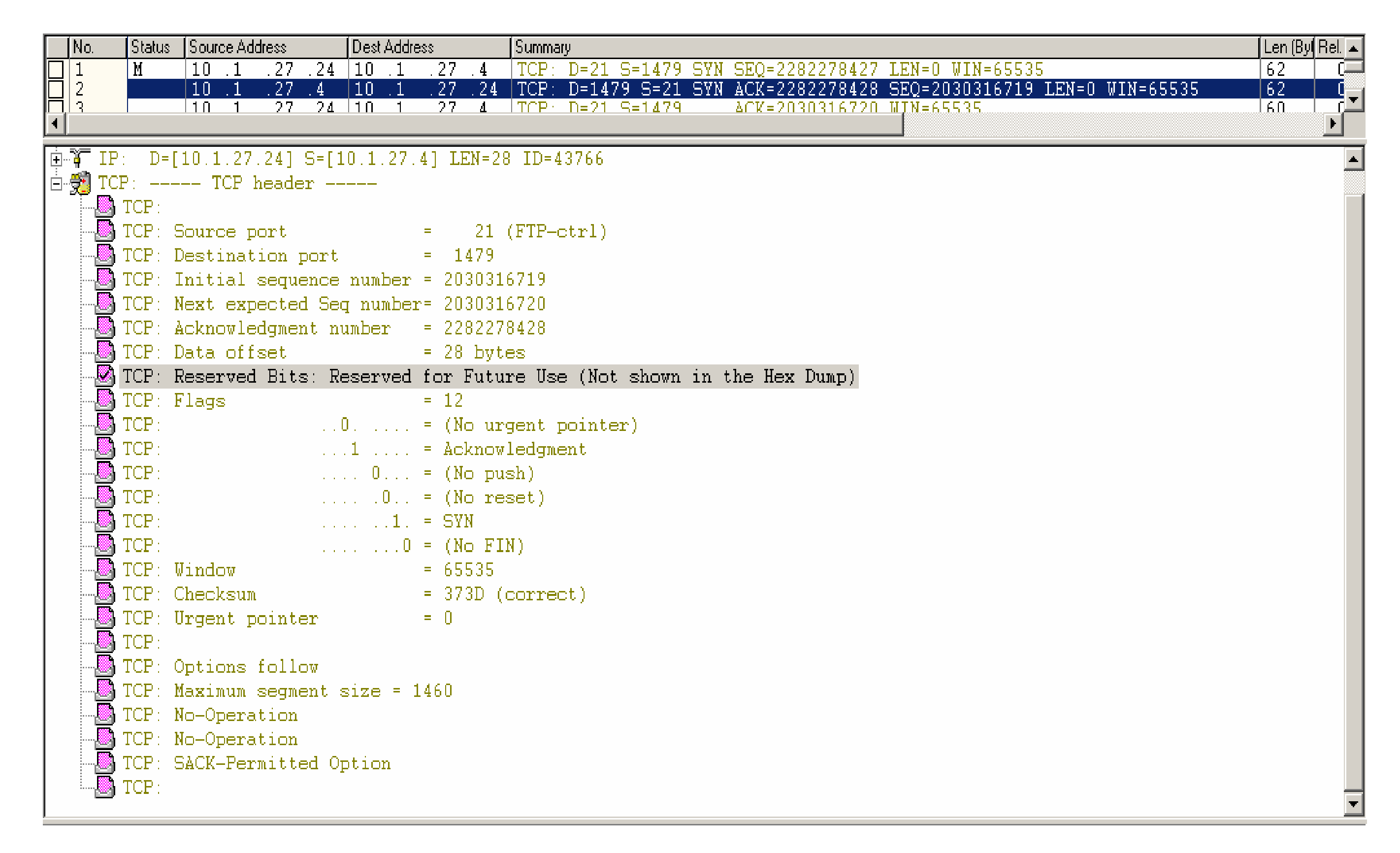


图 13-2

如图 13-3 请求端 208 号机收到 1 号机的信号后，发回信息给 1 号机。标志位 ACK 置为 1，其它标志为都为 0。注意此时 SYN 值为 0，SYN 是标示发起连接的，上两部连接已经完成。

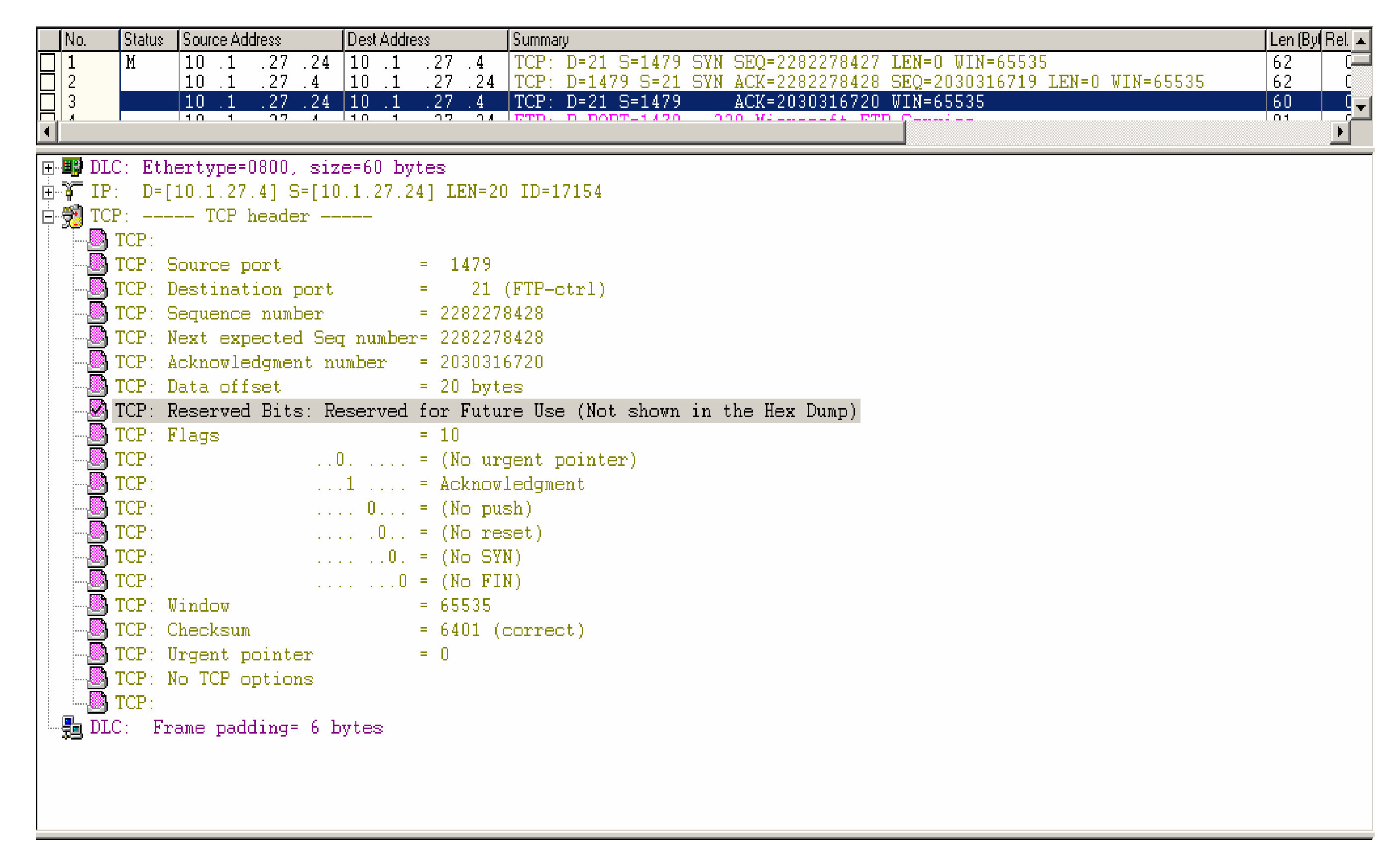


图 13-3

16 位窗口大小：

TCP 的流量控制由连接的每一端通过声明的窗口大小来提供。窗口大小为字节数，起始于确认序号字段指明的值，这个值是接收端正期望接收的字节。窗口大小是一个 16 字节字段，因而窗口大小最大为 65535 字节。

16 位检验和：

检验和覆盖了整个的 TCP 报文段： TCP 首部和 TCP 数据。这是一个强制性的字段，一定是由发端计算和存储，并由收端进行验证。

16 位紧急指针：

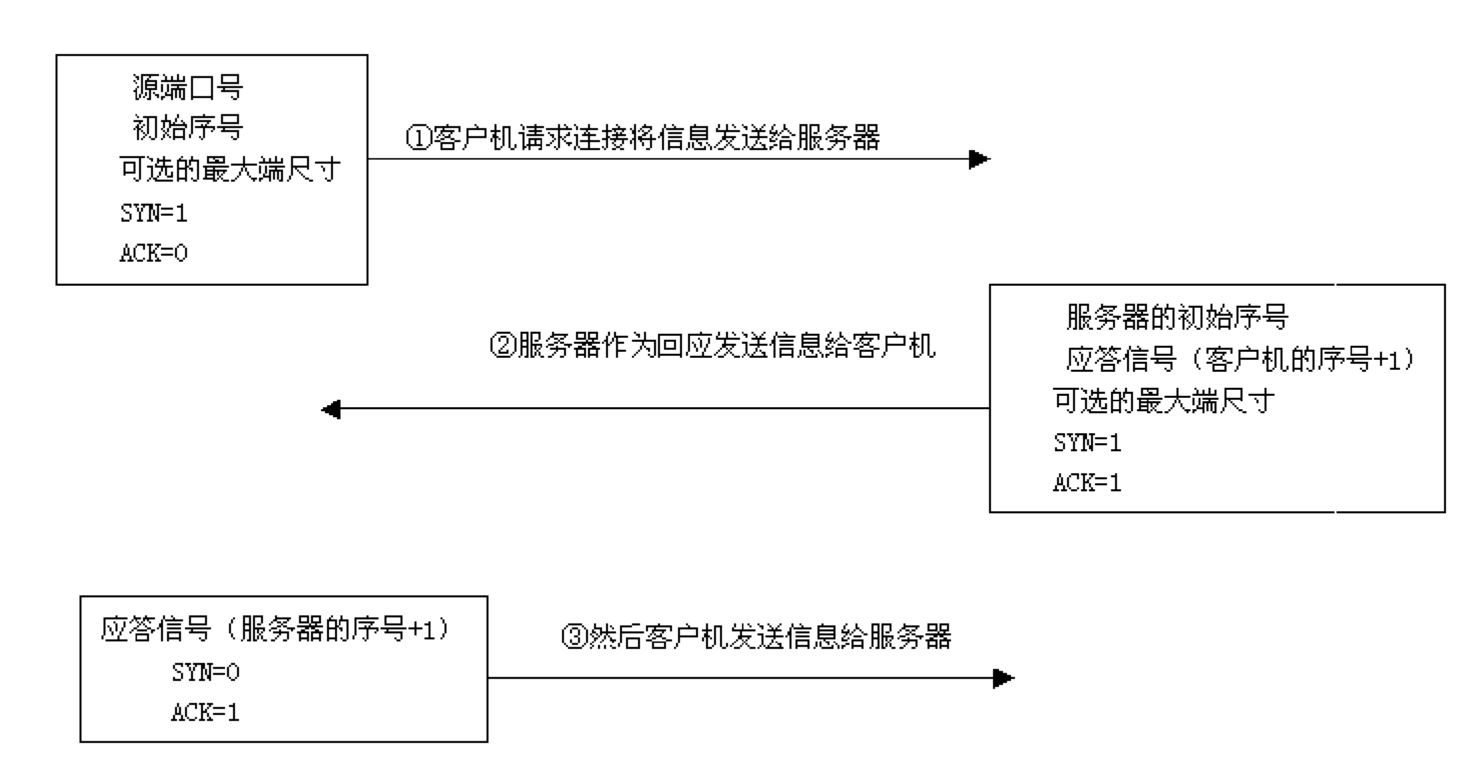
只有当 U R G 标志置 1 时紧急指针才有效。紧急指针是一个正的偏移量，和序号字段中的值相加表示紧急数据最后一个字节的序号。

选项：图 13-1 和图 13-2 有 8 个字节选项，图 13-3 没有选项。最常见的可选字段是最长报文大小，又称为 MSS (Maximum Segment Size)。每个连接方通常都在握手的第一步中指明这个选项。它指明本端所能接收的最大长度的报文段。图 13-1 可以看出 208 号机可以接受的最大字节数为 1460 字节，1460 也是以太网默认的大小，在第三组的数据分析中可以看到数据传送正是以 1460 字节

传送的。

握手小结

上面我们分开讲了三次握手，看着有点散，现在小结一下。



第三组 数据传输

1）下图显示的是 57-60 行的数据

57	10 .1 .27 .4	10 .1 .27 .24	TCP: D=1481 S=2751 ACK=3371471693 SEQ=2725474962 LEN=1460 WIN=65535	1514
58	10 .1 .27 .24	10 .1 .27 .4	TCP: D=2751 S=1481 ACK=2725476422 WIN=65535	60
59	10 .1 .27 .4	10 .1 .27 .24	TCP: D=1481 S=2751 ACK=3371471693 SEQ=2725476422 LEN=1460 WIN=65535	1514
60	10 .1 .27 .4	10 .1 .27 .24	TCP: D=1481 S=2751 ACK=3371471693 SEQ=2725477882 LEN=1460 WIN=65535	1514

图 14

2）解释数据包

这四行数据是数据传输过程中一个发送一个接收的过程。

前文说过，TCP 提供一种面向连接的、可靠的字节流服务。当接收端收到来自发送端的信息时，接受端要发送一条应答信息，表示收到此信息。数据传送时被 TCP 分割成认为最适合发送的数据块。一般以太网在传送时 TCP 将数据分为 1460 字节。也就是说数据在发送方被分成一块一块的发送，接受端收到这些数据后再将它们组合在一起。

57 行显示 1 号机给 2 号机发送了大小为 1514 字节大小的数据，注意我们前文讲过数据发送时是层层加协议头的，1514 字节=14 字节以太网头 + 20 字节 IP 头 + 20 字节 TCP 头 + 1460 字节数据

58 行显示的应答信号 ACK 为：2725476422，这个数是 57 行得 SEQ 序号 2725474962 加上传送的数据 1460，2 号机将这个应答信号发给 1 号机说明已收到发来的数据。

59、60 行显示的是继续传送数据的过程。

这个过程就像我向张三借书，借给我几本我要说：“我已借了你几本了。”，他说：“知道了”。

3）头信息

图 15-1 和图 15-2 分别是 57 行和 58 行的头信息，解释参考第二组。

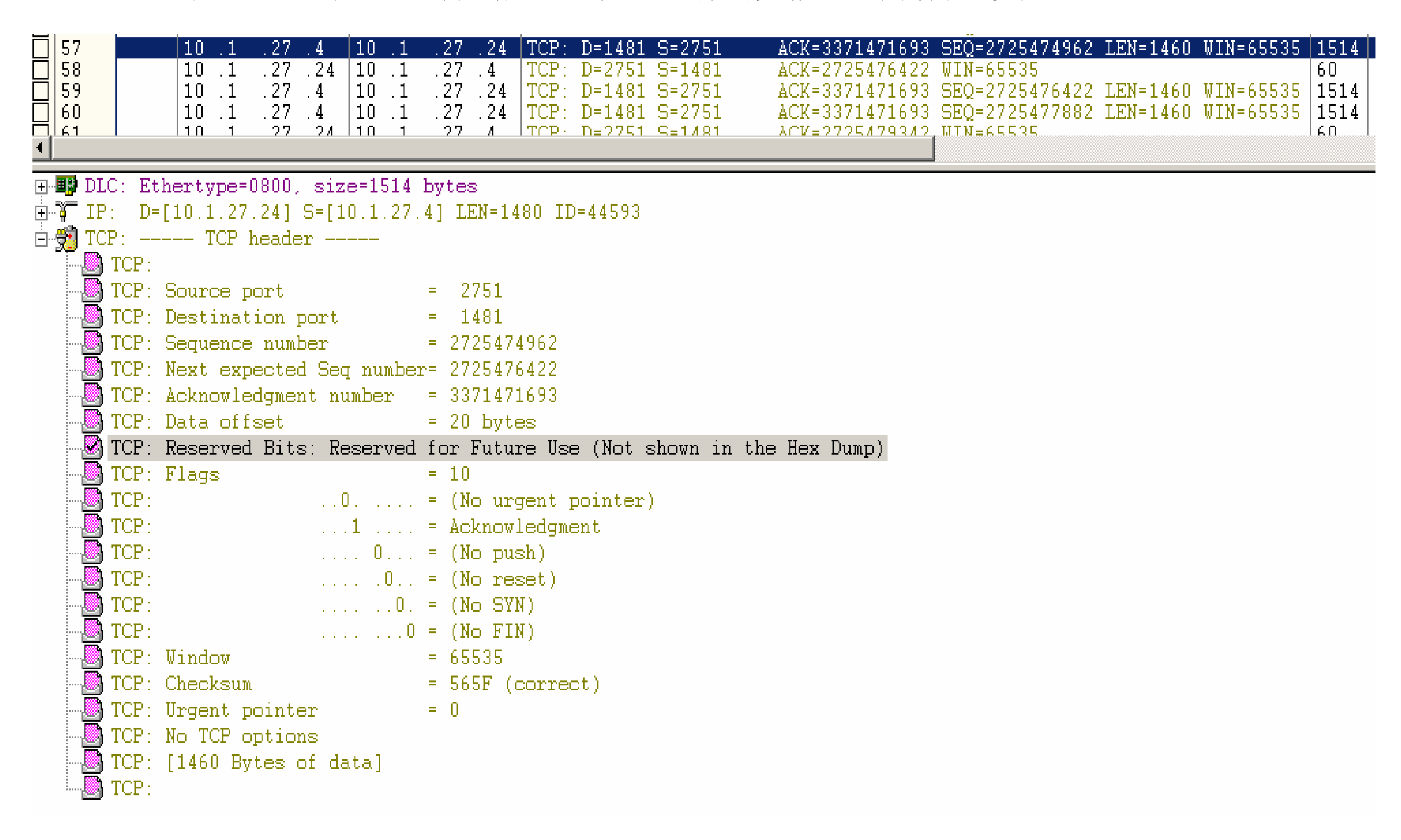


图 15-1

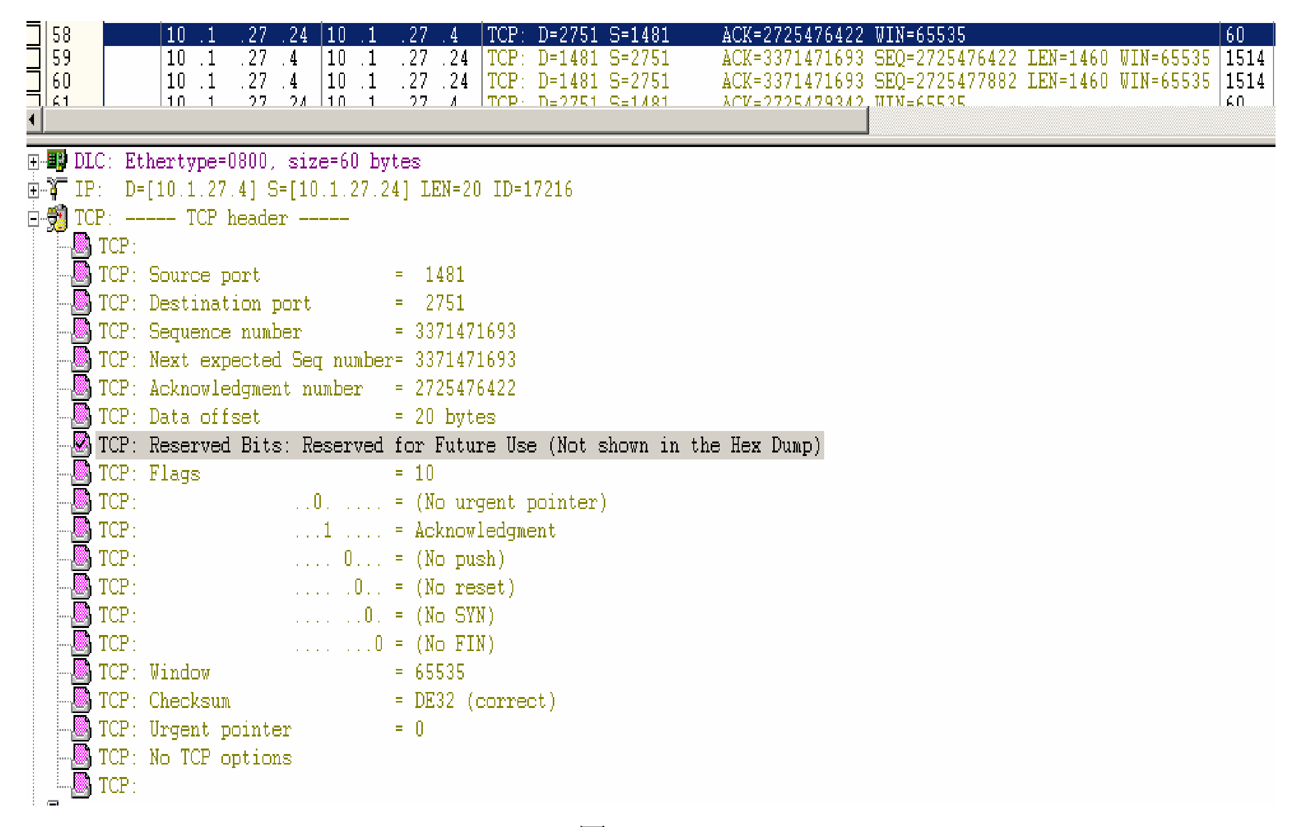


图 15-2

第四组 终止连接

1）下图显示的是 72-76 行的数据(另外扑的数据，非上面)

No.	Source	Destination	Summary	Len
72	10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1756 ACK=2659810009 WIN=64734	60
73	10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1756 FIN ACK=2659810009 SEQ=2188227415 LEN=0 WIN=64734	60
74	10 .1 .27 .4	10 .1 .27 .24	TCP: D=1756 S=21 ACK=2188227416 WIN=65373	60
75	10 .1 .27 .4	10 .1 .27 .24	TCP: D=1756 S=21 FIN ACK=2188227416 SEQ=2659810009 LEN=0 WIN=65373	60
76	10 .1 .27 .24	10 .1 .27 .4	TCP: D=21 S=1756 ACK=2659810010 WIN=64734	60

图 16

2）解释数据包

72-76 是两机通讯完关闭的过程。

建立一个连接需要三次握手，而终止一个连接要经过 4 次握手。这是因为一个 TCP 连接是全双工（即数据在两个方向上能同时传递），每个方向必须单独地进行关闭。4 次握手实际上就是双方单独关闭的过程。

本例文件下载完后，关闭浏览器终止了与服务器的连接图 16 的 72-76 行显示的就是终止连接所经过 4 次握手过程。

73 行数据显示的是关闭浏览器后，如图 17-1 所示 2 号机将 FIN 置 1 连同序号(SEQ)2188227415 发给 1 号机请求终止连接。

```
73   10 .1 .27 .24  10 .1 .27 .4   TCP: D=21 S=1756 FIN ACK=2659810009 SEQ=2188227415 LEN=0 WIN=64734
74   10 .1 .27 .4   10 .1 .27 .24  TCP: D=1756 S=21     ACK=2188227416 WIN=65373

IP:
TCP: ----- TCP header -----
TCP:
TCP: Source port              =  1756
TCP: Destination port         =    21 (FTP-ctrl)
TCP: Sequence number          = 2188227415
TCP: Next expected Seq number= 2188227416
TCP: Acknowledgment number    = 2659810009
TCP: Data offset              = 20 bytes
TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
TCP: Flags                    = 11
TCP:            ..0. .... = (No urgent pointer)
TCP:            ...1 .... = Acknowledgment
TCP:            .... 0... = (No push)
TCP:            .... .0.. = (No reset)
TCP:            .... ..0. = (No SYN)
TCP:            .... ...1 = FIN
TCP: Window                   = 64734
TCP: Checksum                 = 12BE (correct)
TCP: Urgent pointer           = 0
TCP: No TCP options
TCP:
```

图 17-1

74 行数据和图 17-2 显示 1 号机收到 FIN 关闭请求后，发回一个确认，并将应答信号设置为收到序号加 1，这样就终止了这个方向的传输。

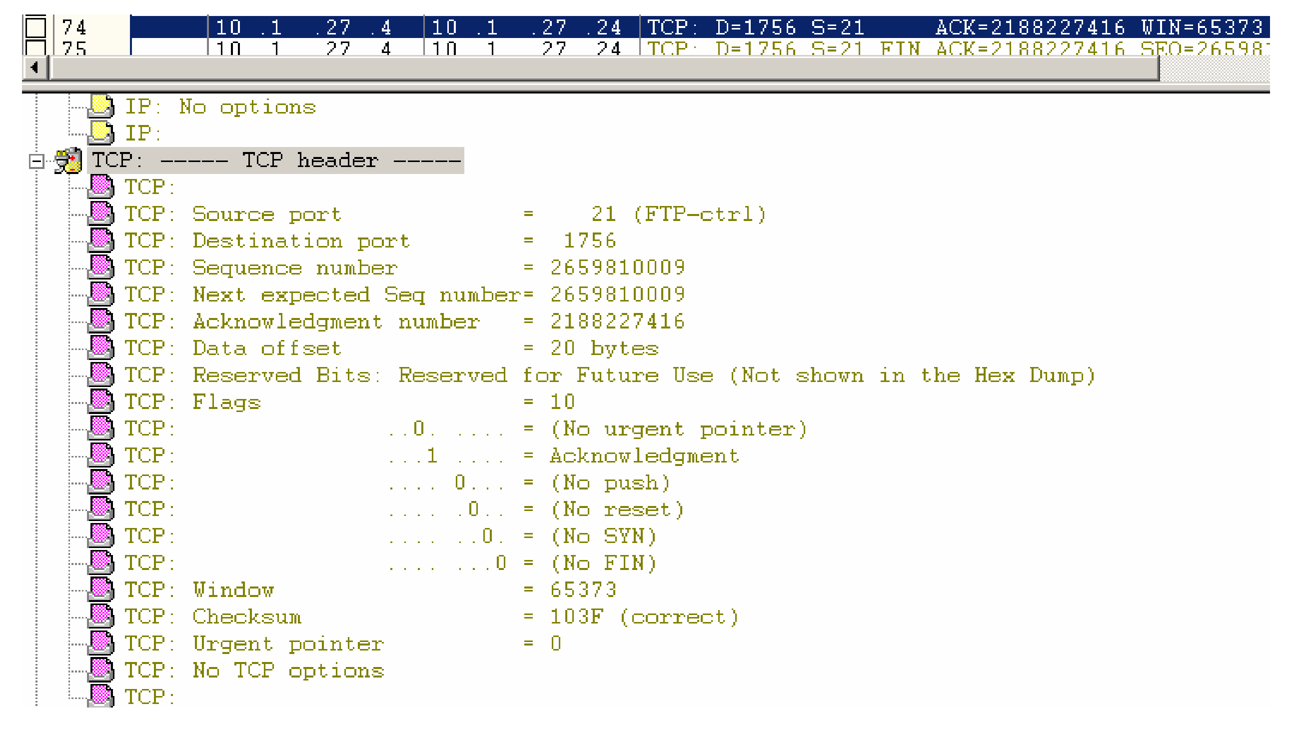


图 17-2

75 行数据和图 17-3 显示 1 号机将 FIN 置 1 连同序号(SEQ)2659810009 发给 2 号机请求终止连接。

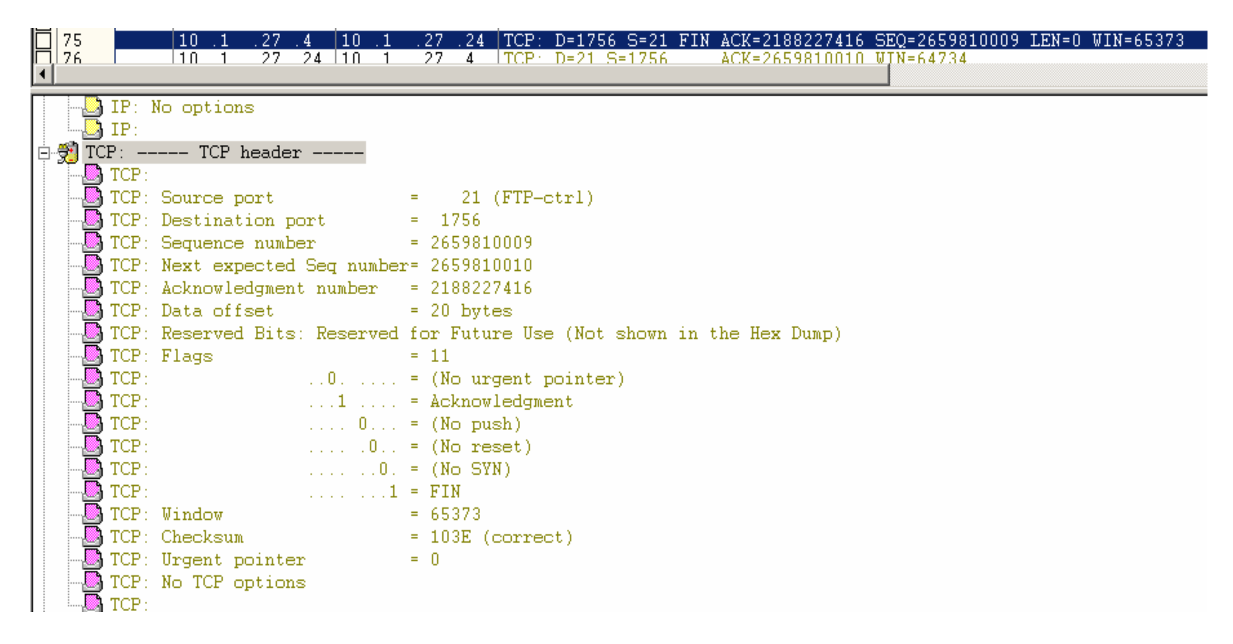


图 17-3

76 行数据和图 17-4 显示 208 号机收到 FIN 关闭请求后，发回一个确认，并将应答信号设置为收到序号加 1，至此 TCP 连接彻底关闭。

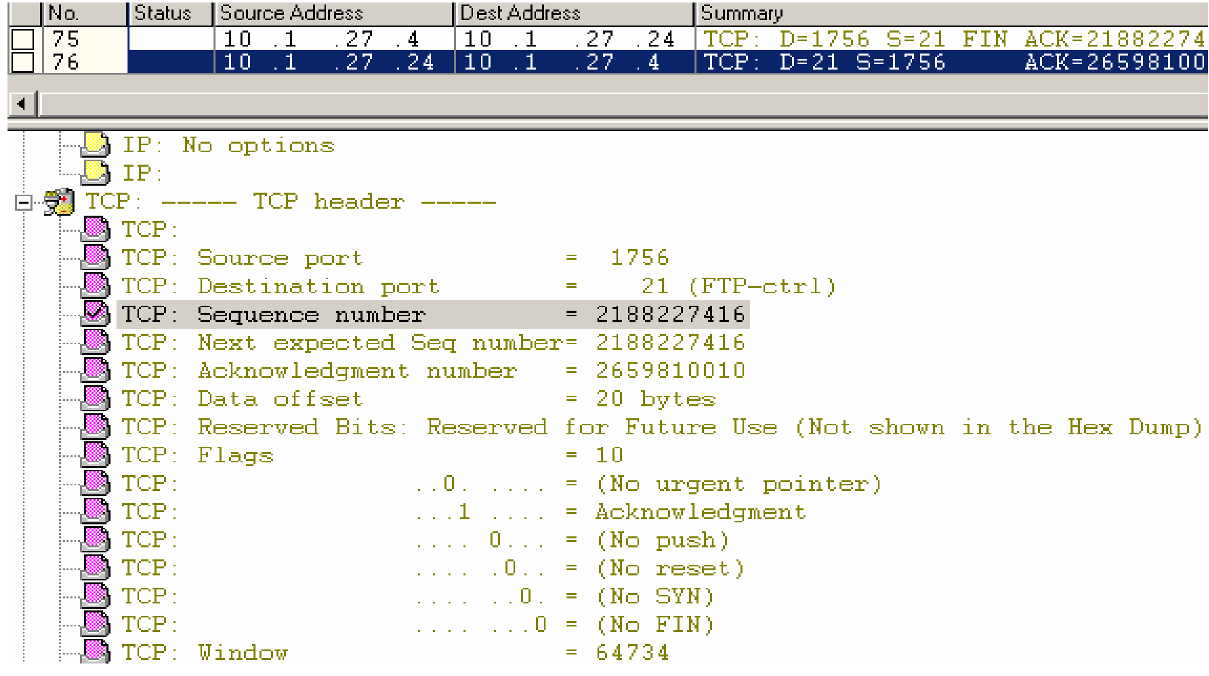


图 17-4

五、扫描实例

下面我们再举个 ping 的实例，测试某台计算机是否通，最常用的命令就是 ping 命令。Ping 一台计算机，出现如图 18 所示界面就是通，出现如图 19 所示界面就是不通，不通有两种情况，一是该计算机不存在或没接网线，二是该计算机安装了防火墙并设置为不允许 ping。如何区别这两种情况呢？下面还是利用 Sniffer Pro 跟踪上述情况。

```
C:\>ping 10.1.27.4

Pinging 10.1.27.4 with 32 bytes of data:

Reply from 10.1.27.4: bytes=32 time<1ms TTL=64
Reply from 10.1.27.4: bytes=32 time<1ms TTL=64
Reply from 10.1.27.4: bytes=32 time<1ms TTL=64
Reply from 10.1.27.4: bytes=32 time<1ms TTL=64

Ping statistics for 10.1.27.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

图 18

```
C:\>ping 10.1.27.4

Pinging 10.1.27.4 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.1.27.4:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 19

如图 20 是 ping 通的情况。

如图 21 是 ping 不通该计算机不存在的情况。从图可以看出 ARP 请求没有回应。

如图 22 是 ping 不通，该计算机存在但安装了防火墙的情况。从图可以看出 ARP 请求有回应。但 ICMP 请求没回应。

从分析可以看出虽然后两种情况的表面现象是一样的，但实质确是截然相反的。通过头信息可以清楚的看出 PING 是 ICMP 协议来完成的，通讯过程是在第三层完成的，没有用到第四层的 TCP 协议。

No.	Status	Source Address	Dest Address	Summary
1	M	10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo
2		10 .1 .27 .4	10 .1 .27 .24	ICMP: Echo reply
3		10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo
4		10 .1 .27 .4	10 .1 .27 .24	ICMP: Echo reply
5		10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo
6		10 .1 .27 .4	10 .1 .27 .24	ICMP: Echo reply
7		10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo
8		10 .1 .27 .4	10 .1 .27 .24	ICMP: Echo reply

```
DLC: Ethertype=0800, size=74 bytes
IP:  D=[10.1.27.4] S=[10.1.27.24] LEN=40 ID=45186
ICMP: ----- ICMP header -----
    ICMP:
    ICMP: Type = 8 (Echo)
    ICMP: Code = 0
    ICMP: Checksum = EF5B (correct)
    ICMP: Identifier = 1024
    ICMP: Sequence number = 23040
    ICMP: [32 bytes of data]
    ICMP:
    ICMP: [Normal end of "ICMP header".]
    ICMP:
```

图 20

No.	Status	Source Address	Dest Address	Summary
1	M	10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo
2		10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo
3		10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo
4		10 .1 .27 .24	10 .1 .27 .4	ICMP: Echo

```
DLC: Ethertype=0800, size=74 bytes
IP:  D=[10.1.27.4] S=[10.1.27.24] LEN=40 ID=45617
ICMP: ----- ICMP header -----
    ICMP:
    ICMP: Type = 8 (Echo)
    ICMP: Code = 0
    ICMP: Checksum = EB5B (correct)
    ICMP: Identifier = 1024
    ICMP: Sequence number = 24064
    ICMP: [32 bytes of data]
    ICMP:
    ICMP: [Normal end of "ICMP header".]
    ICMP:

00000000: 00 13 72 aa fe 70 00 01 6c 8b dd 02 08 00 45 00  ..r  p..l燉...E.
00000010: 00 3c b2 31 00 00 80 01 3e 72 0a 01 1b 18 0a 01  .<?..▌.>r......
00000020: 1b 04 08 00 eb 5b 04 00 5e 00 61 62 63 64 65 66  ....隱..^.abcdef
00000030: 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmnopqrstuv
00000040: 77 61 62 63 64 65 66 67 68 69                    wabcdefghi
```

图 21

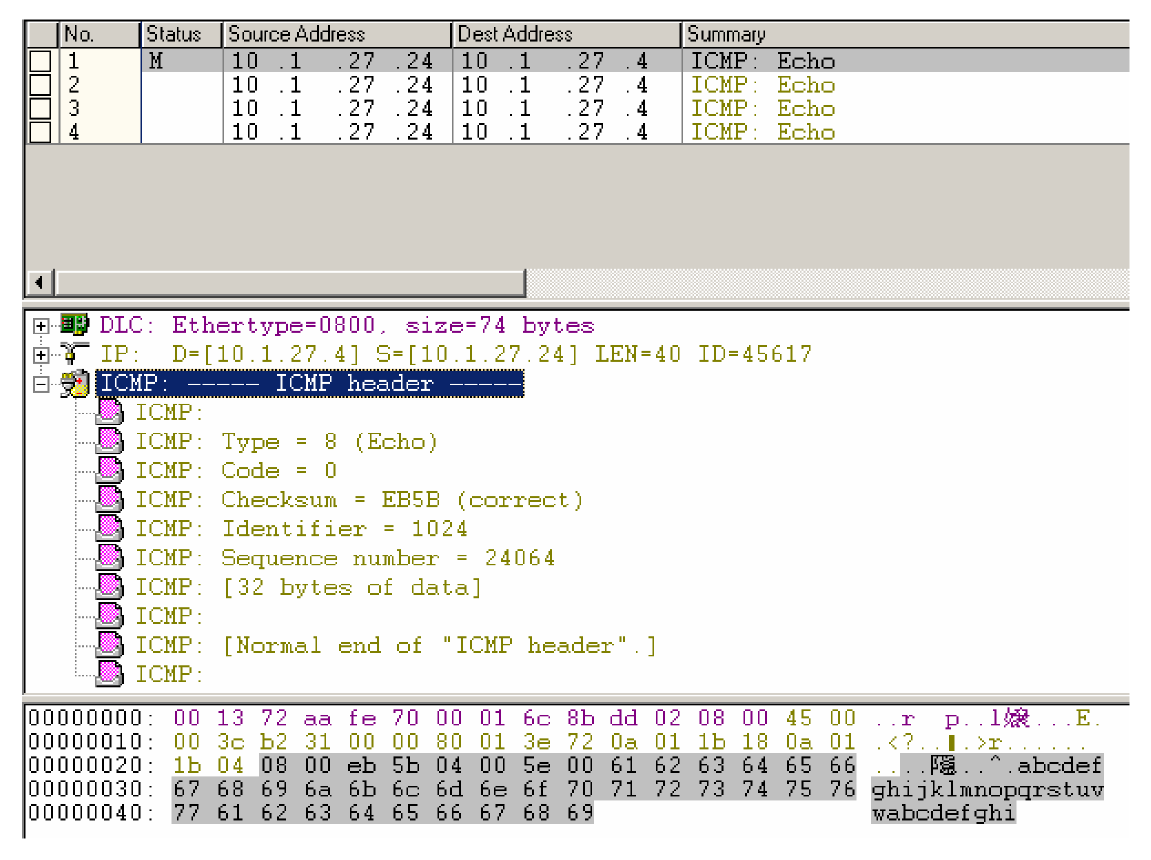


图 22

六、后记

本文不是个教程，许多问题都没有涉及到，比如 TCP 重发、IP 分解、路由等，只是提出个学习思路，希望能起到抛砖引玉的作用。TCP/IP 协议族是非常复杂的，但只要理解了还是不难学的。最后向感兴趣的朋友提个问题：分别 telnet 三台机器，一台正常 23 端口开放，一台网是通的但 23 端口没开放，另外一台是不存在的。用我们学过的方法跟踪一下，比较三个的不同。其实这就是用 TCP 扫描判断对方机器是否在线的一种方法。

sniffer 原理

现在人们谈到黑客攻击，一般所指的都是以主动方式进行的，例如利用漏洞或者猜测系统密码的方式对系统进行攻击。但是其实还有一类危害非常大的被动攻击方式往往为大家所忽视，那就是利用 Sniffer 进行嗅探攻击。

Sniffer，中文可以翻译为嗅探器，是一种威胁性极大的被动攻击工具。使用这种工具，可以监视网络的状态、数据流动情况以及网络上传输的信息。当信息以明文的形式在网络上传输时，便可以使用网络监听的方式来进行攻击。将网络接口设置在监听模式，便可以将网上传输的源源不断的信息截获。黑客们常常用它来截获用户的口令。据说某个骨干网络的路由器曾经被黑客攻人，并嗅探到大量的用户口令。本文将详细介绍 Sniffer 的原理和应用。

一、Sniffer 原理

1. 网络技术与设备简介

在讲述 Sni 计 er 的概念之前，首先需要讲述局域网设备的一些基本概念。

数据在网络上是以很小的称为帧(Frame)的单位传输的，帧由几部分组成，不同的部分执行不同的功能。帧通过特定的称为网络驱动程序的软件进行成型，然后通过网卡发送到网线上，通过网线到达它们的目的机器，在目的机器的一端执行相反的过程。接收端机器的以太网卡捕获到这些帧，并告诉操作系统帧已到达，然后对其进行存储。就是在这个传输和接收的过程中，嗅探器会带来安全方面的问题。

每一个在局域网（LAN）上的工作站都有其硬件地址，这些地址惟一地表示了网络上的机器（这一点与 Internet 地址系统比较相似）。当用户发送一个数据包时，这些数据包就会发送到 LAN 上所有可用的机器。

在一般情况下，网络上所有的机器都可以“听”到通过的流量，但对不属于自己的数据包则不予响应(换句话说，工作站 A 不会捕获属于工作站 B 的数据，而是简单地忽略这些数据)。如果某个工作站的网络接口处于混杂模式（关于混杂模式的概念会在后面解释），那么它就可以捕获网络上所有的数据包和帧。

2. 网络监听原理

Sniffor 程序是一种利用以太网的特性把网络适配卡（NIC，一般为以太同卡）置为杂乱（promiscuous）模式状态的工具，一旦同卡设置为这种模式，它就能接收传输在网络上的每一个信息包。

普通的情况下，阿卡只接收和自己的地址有关的信息包，即传输到本地主机的信息包。要使 Sniffer 能接收并处理这种方式的信息，系统需要支持 BPF，Linux 下需要支持 SOCKET 一 PACKET。但一般情况下，网络硬件和 TCP／IP 堆栈不支持接收或者发送与本地计算机无关的数据包，所以，为了绕过标准的 TCP／IP 堆栈，网卡就必须设置为我们刚开始讲的混杂模式。一般情况下，要激活这种方式，内核必须支持这种伪设备 Bpfilter，而且需要 root 权限来运行这种程序，所以 sniffer 需要 root 身份安装，如果只是以本地用户的身份进人了系统，那么不可能唤探到 root 的密码，因为不能运行 Sniffer。

基于 Sniffer 这样的模式，可以分析各种信息包并描述出网络的结构和使用的机器，由于它接收任何一个在同一网段上传输的数据包，所以也就存在着捕获密码、各种信息、秘密文档等一些没有加密的信息的可能性。这成为黑客们常用的扩大战果的方法，用来夺取其他主机的控制权

3 Snifffer 的分类

Sniffer 分为软件和硬件两种，软件的 Sniffer 有 NetXray、Packetboy、Net monitor 等，其优点是物美价廉，易于学习使用，同时也易于交流；缺点是无法抓取网络上所有的传输，某些情况下也就无法真正了解网络的故障和运行情况。硬件的 Sniffer 通常称为协议分析仪，一般都是商业性的，价格也比较贵。

实际上本文中所讲的 Sniffer 指的是软件。它把包抓取下来，然后打开并查看其中的内容，可以得到密码等。Sniffer 只能抓取一个物理网段内的包，就是说，你和监听的目标中间不能有路由或其他屏蔽广播包的设备，这一点很重要。所以，对一般拨号上网的用户来说，是不可能利用 Sniffer 来窃听到其他人的通信内容的。

4. 网络监听的目的

当一个黑客成功地攻陷了一台主机，并拿到了 root 权限，而且还想利用这台主机去攻击同一网段上的其他主机时，他就会在这台主机上安装 Sniffer 软件，对以太网设备上传送的数据包进行侦听，从而发现感兴趣的包。如果发现符合条件的包，就把它存到一个 LOg 文件中去。通常设置的这些条件是包含字“username”或“password”的包，这样的包里面通常有黑客感兴趣的密码之类的东西。一旦黑客截获得了某台主机的密码，他就会立刻进人这台主机。

如果 Sniffer 运行在路由器上或有路由功能的主机上，就能对大量的数据进行监控，因为所有进出网络的数据包都要经过路由器。

Sniffer 属于第 M 层次的攻击。就是说，只有在攻击者已经进入了目标系统的情况下，才能使用 Sniffer 这种攻击手段，以便得到更多的信息。

Sniffer 除了能得到口令或用户名外，还能得到更多的其他信息，比如一个重要的信息、在网上传送的金融信息等等。Sniffer 几乎能得到任何在以大网上传送的数据包。

Sniffer 是一种比较复杂的攻击手段，一般只有黑客老手才有能力使用它，而对于一个网络新手来说，即使在一台主机上成功地编译并运行了 Sniffer，一般也不会得到什么有用的信息，因为通常网络上的信息流量是相当大的，如果不加选择地接收所有的包，然后从中找到所需要的信息非常困难；而且，如果长时间进行监听，还有可能把放置 Snifkr 的机器的硬盘撑爆。

5. 一个简单的 Sniffer 程序

下面是一个非常简单的 C 程序，它可以完成一般的监听功能，/＊＊/ 中的内容是本文的注解。

```
/*下面是包含进行调用系统和网络函数的头文件*/
#include〈stdio.h〉
#include〈sys/socket.h〉
#include〈netinet/in.h〉
#include〈arpa/inet.h〉
/*下面是 IP 和 TCP 包头结构*/
struct IP{
 unsigned int ip_length:4;
 /*定义 IP 头的长度*/
 unsigned int ip_version:4;
 /*IP 版本，Ipv4 */
 unsigned char ip_tos;
 /*服务类型*/
```

```
 unsigned short
ip_total_length; /*IP 数据包的总长度*/
 unsigned short ip_id;
 /*鉴定城*/
 unsigned short ip_flags;
 /*IP 标志 */
 unsigned char ip_ttl;
 /*IP 包的存活期*/
 unsigned char ip_protocol;
 /*IP 上层的协议*/
 unsigned short ip_cksum;
 /*IP 头校验和*/
unsigned int ip_source ;
/*源 IP 地址*/
unsigned int ip_source;
/*目的 IP 地址*/
 };
struct tcp{
unsigned short tcp_source_port;
/*定义 TCP 源端口*/
 unsigned short tcp_dest_port;
 /*TCP 目的端口*/
 unsigned short tcp_seqno;
 /*TC P 序列号*/
unsigned int tcp_ackno;
/*发送者期望的下一个序列号*/
unsigned int tcp_res1:4;
/*下面几个是 TCP 标志*/
tcp_hlen:4
tcp_fin:1,
tcp_syn:1,
tcp_rst:1,
tcp_psh:1,
tcp_ack:1,
tcp_urg:1,
tcp_res2:2;
unsignd short tcp_winsize; /*能接收的最大字节数*/
 unsigned short tcp_cksum;
 /* TCP 校验和*/
 unsigned short tcp_urgent;
 /* 紧急事件标志*/
};
/*主函数*/
int main()
```

```
{
int sock,bytes_recieved,fromlen;
char buffer[65535];
struct sockaddr_in from;
 /*定义 socket 结构*/
struct ip ip;
 /*定义 IP 和 TCP*/
struct tcp *tcp;
 sock=socket(AF_INET,SOCK_RAW,IPPROTO_TCP);
/* 上面是建立 socket 连接，第一个参数是地址族类型，用 INTERNET 类型*/
/* 第二个参数是 socket 类型，这里用了 SOCK_RAW，它可以绕过传输层*/
/* 直接访问 IP 层的包，为了调用 SOCK_RAW,需要有 root 权限*/
/* 第三个参数是协议，选 IPPROTO_TCP 指定了接收 TCP 层的内容*/
while(1)
/*建立一个死循环，不停的接收网络信息*/
{
fromlen=sizeof from;
       bytes_recieved=recvfrom(sock,buffer,sizeofbuffer,0,(struct sockaddr *)&from,&fromlen);
/*上面这个函数是从建立的 socket 连接中接收数据*/
/*因为 recvfrom()需要一个 sockaddr 数据类型，所以我们用了一个强制类型转换*/
print("\nBytes received ::: %5d\n",bytes_recieved);
/*显示出接收的数据字节数*/
printf("source address ::: %s\n",inet_ntoa(from.sin_addr));
/*显示出源地址*/
ip=(struct ip *)buffer;
/*把接收的数据转化为我们预先定义的结构，便于查看*/
printf("IP header length ::: %d\n",ip->ip_length);
/*显示 IP 头的长度*/
print("Protocol ::: %d\n",ip->ip_protocol);
/*显示协议类型，6 是 TCP，17 是 UDP*/
tcp=(struct tcp *)(buffer + (4*ip->ip_iplength));
/*上面这名需要详细解释一下，因为接收的包头数据中，IP 头的大小是固定的 4 字节*/
/*所以我用 IP 长度乘以 4,指向 TCP 头部分*/
printf("Source port ::: %d\n",ntohs(tcp->tcp_source_port); /*显示出端口*/
printf("Dest prot ::: %d\n",ntohs(tcp->tcp_dest_port));/*显示出目标端口*/
```

以上这个 C 程序是为了说明 Sniffer 的接收原理而列举的一个

Sniffer Pro的基本使用和实例(一)

运行环境及安装

Sniffer Pro 可运行在局域网的任何一台机器上，如果是练习使用，网络连接最好用 Hub 且在一个子网，这样能抓到连到 Hub 上每台机器传输的包。

本文用的版本是 4.6，Sniffer Pro软件的获取可在www.baidu.com或www.google.com 中输入 Sniffer Pro 4.6，查找相应的下载站点来下载。 该版本是不要序列号的。

安装非常简单，setup 后一路确定即可，第一次运行时需要选择你的网卡。

最好在 win2000 下运行，在 win2003 下运行网络流量表有问题。

常用功能介绍

1、Dashboard （网络流量表）

点击图 1 中①所指的图标，出现三个表，第一个表显示的是网络的使用率（Utilization），第二个表显示的是网络的每秒钟通过的包数量（Packets），第三个表显示的是网络的每秒错误率（Errors）。通过这三个表可以直观的观察到网络的使用情况，红色部分显示的是根据网络要求设置的上限。

选择图 1 中②所指的选项将显示如图 2 所示的更为详细的网络相关数据的曲线图。每个子项的含义无需多言，下面介绍一下测试网络速度中的几个常用单位。

在 TCP/IP 协议中，数据被分成若干个包（Packets）进行传输，包的大小跟操作系统和网络带宽都有关系，一般为 64、128、256、512、1024、1460 等，包的单位是字节。

很多初学者对 Kbps、KB、Mbps 等单位不太明白，B 和 b 分别代表 Bytes(字节) 和 bits（比特），1 比特就是 0 或 1。1 Byte = 8 bits 。

1Mbps (megabits per second 兆比特每秒)，亦即 1 x 1024 / 8 = 128KB/sec（字节/秒），我们常用的 ADSL 下行 512K 指的是每秒 512K 比特(Kb)， 也就是每秒 512/8=64K 字节（KB）

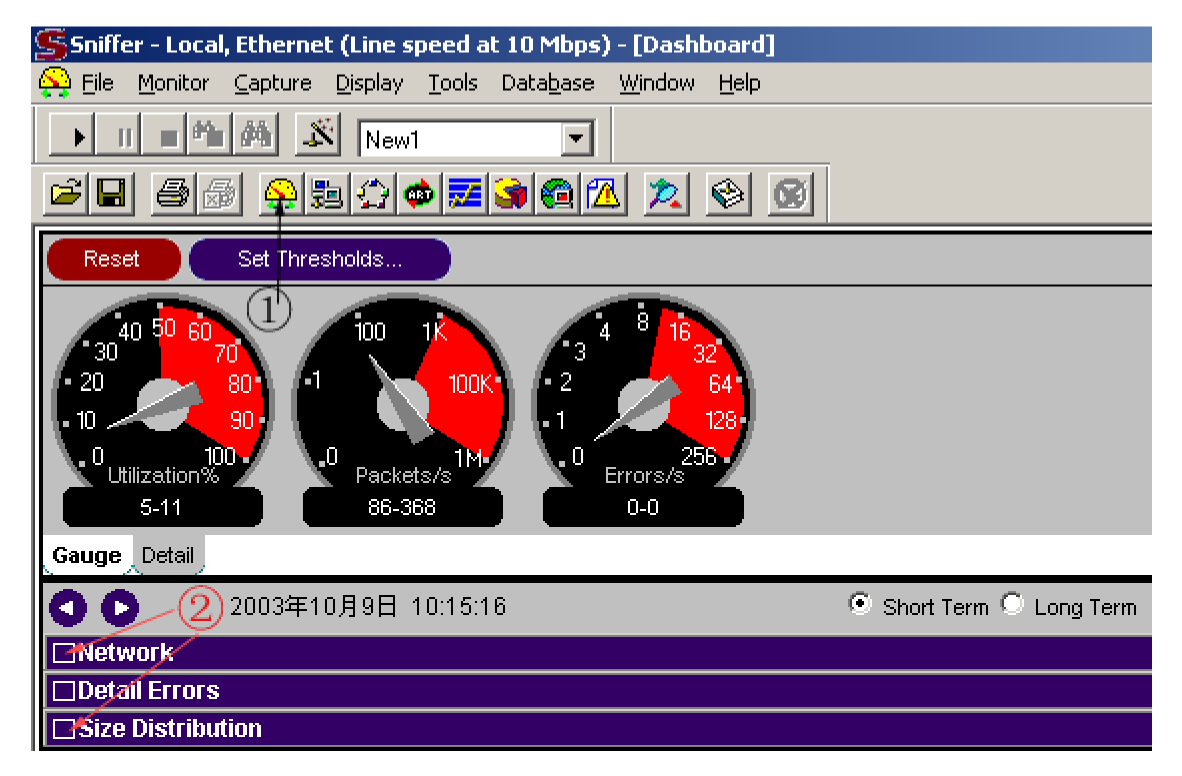


图 1

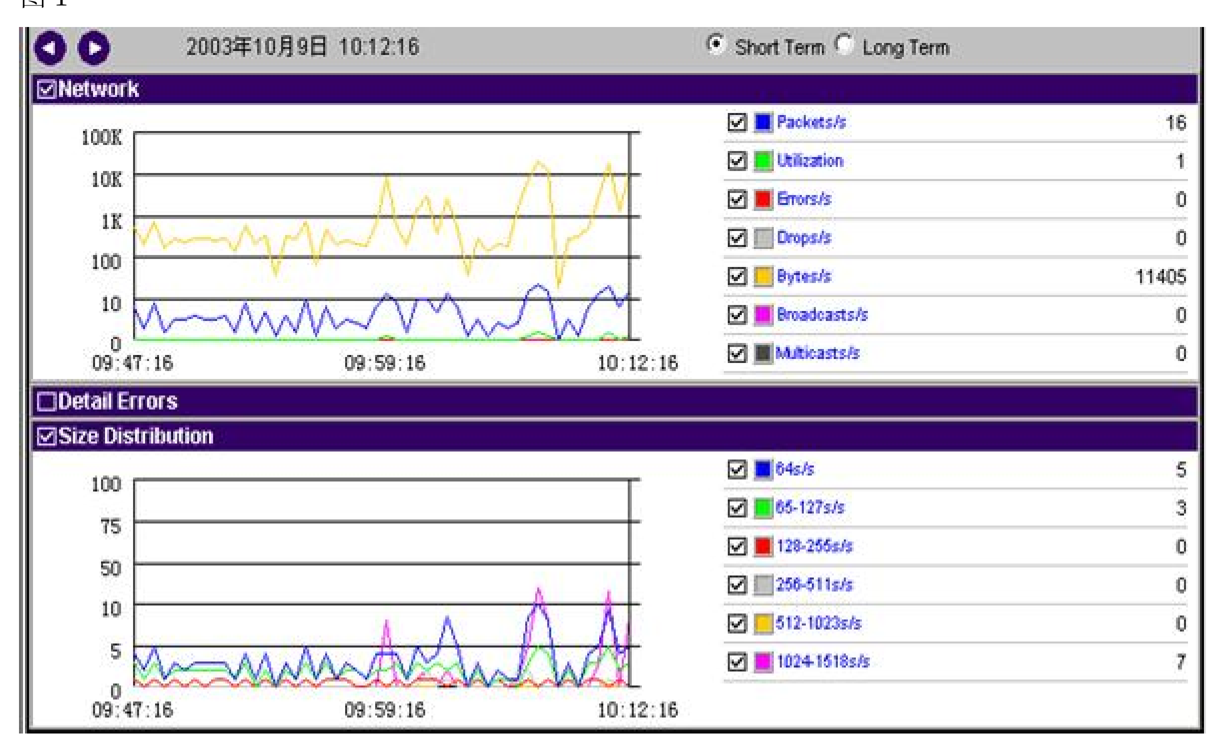


图 2

2、Host table（主机列表）

如图 3 所示，点击图 3 中①所指的图标，出现图中显示的界面，选择图中②所指的 IP 选项，界面中出现的是所有在线的本网主机地址及连到外网的外网服务器地址，此时想看看 192.168.113.88 这台机器的上网情况，只需如图中③所示单击该地址出现图 4 界面。

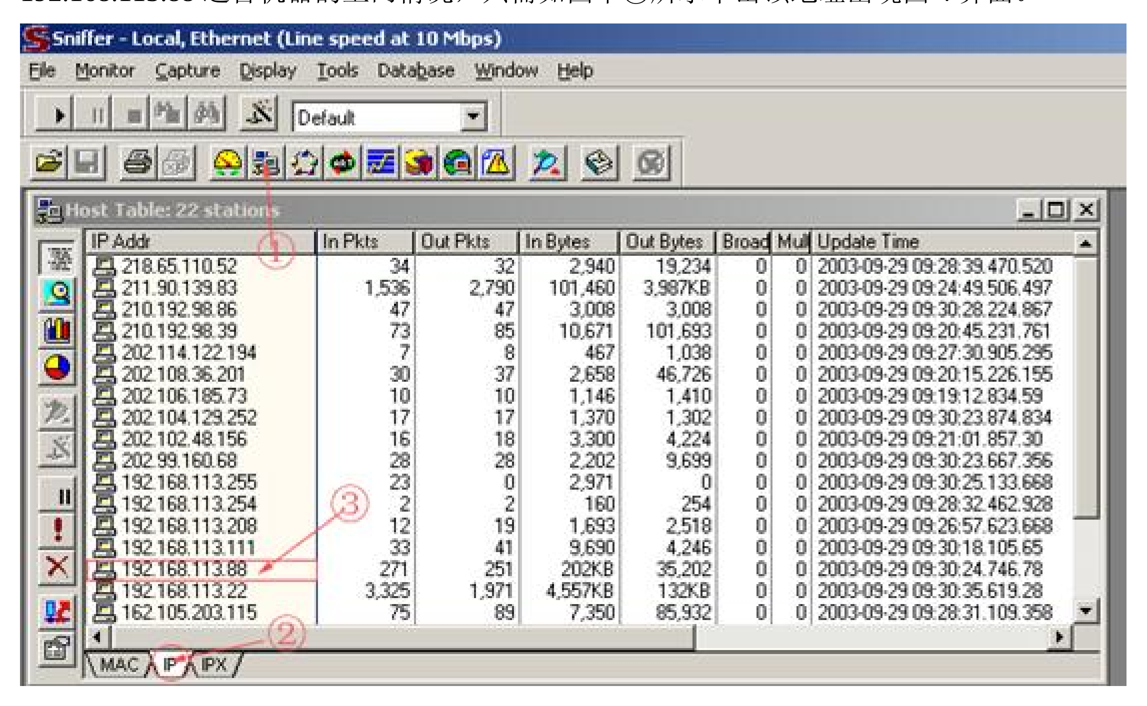


图 3

图 4 中清楚地显示出该机器连接的地址。点击左栏中其它的图标都会弹出该机器连接情况的相关数据的界面。

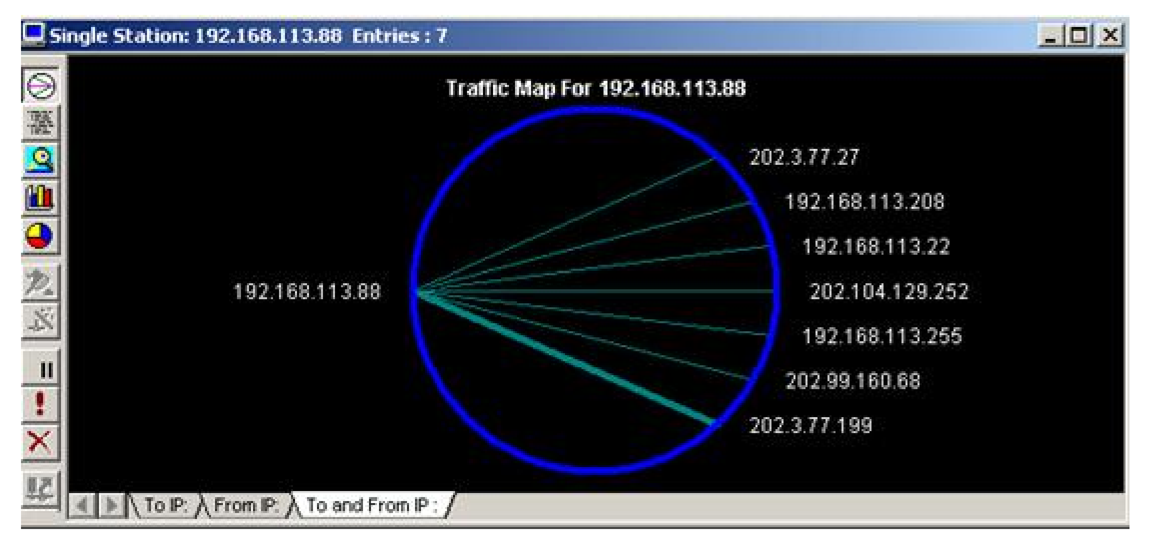
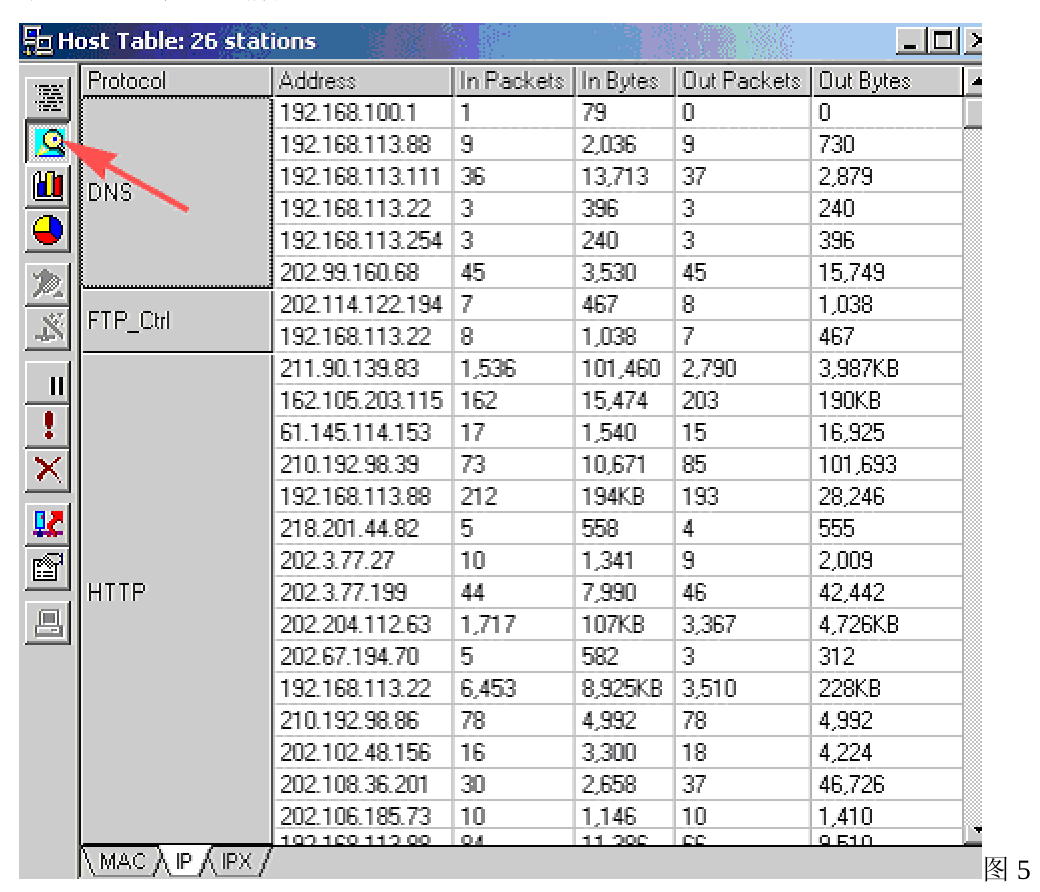


图 4

3、Detail（协议列表）

点击图 5 所示的“Detail”图标，图中显示的是整个网络中的协议分布情况，可清楚地看出哪台机器运行了那些协议。注意，此时是在图 3 的界面上点击的，如果在图 4 的界面上点击显示的是那台机器的情况。



Host Table: 26 stations

Protocol	Address	In Packets	In Bytes	Out Packets	Out Bytes
DNS	192.168.100.1	1	79	0	0
	192.168.113.88	9	2,036	9	730
	192.168.113.111	36	13,713	37	2,879
	192.168.113.22	3	396	3	240
	192.168.113.254	3	240	3	396
	202.99.160.68	45	3,530	45	15,749
FTP_Ctrl	202.114.122.194	7	467	8	1,038
	192.168.113.22	8	1,038	7	467
HTTP	211.90.139.83	1,536	101,460	2,790	3,987KB
	162.105.203.115	162	15,474	203	190KB
	61.145.114.153	17	1,540	15	16,925
	210.192.98.39	73	10,671	85	101,693
	192.168.113.88	212	194KB	193	28,246
	218.201.44.82	5	558	4	555
	202.3.77.27	10	1,341	9	2,009
	202.3.77.199	44	7,990	46	42,442
	202.204.112.63	1,717	107KB	3,367	4,726KB
	202.67.194.70	5	582	3	312
	192.168.113.22	6,453	8,925KB	3,510	228KB
	210.192.98.86	78	4,992	78	4,992
	202.102.48.156	16	3,300	18	4,224
	202.108.36.201	30	2,658	37	46,726
	202.106.185.73	10	1,146	10	1,410

图 5

4、Bar（流量列表）

点击图 6 所示的“Bar”图标，图中显示的是整个网络中的机器所用带宽前 10 名的情况。显示方式是柱状图，图 7 显示的内容与图 6 相同，只是显示方式是饼图。

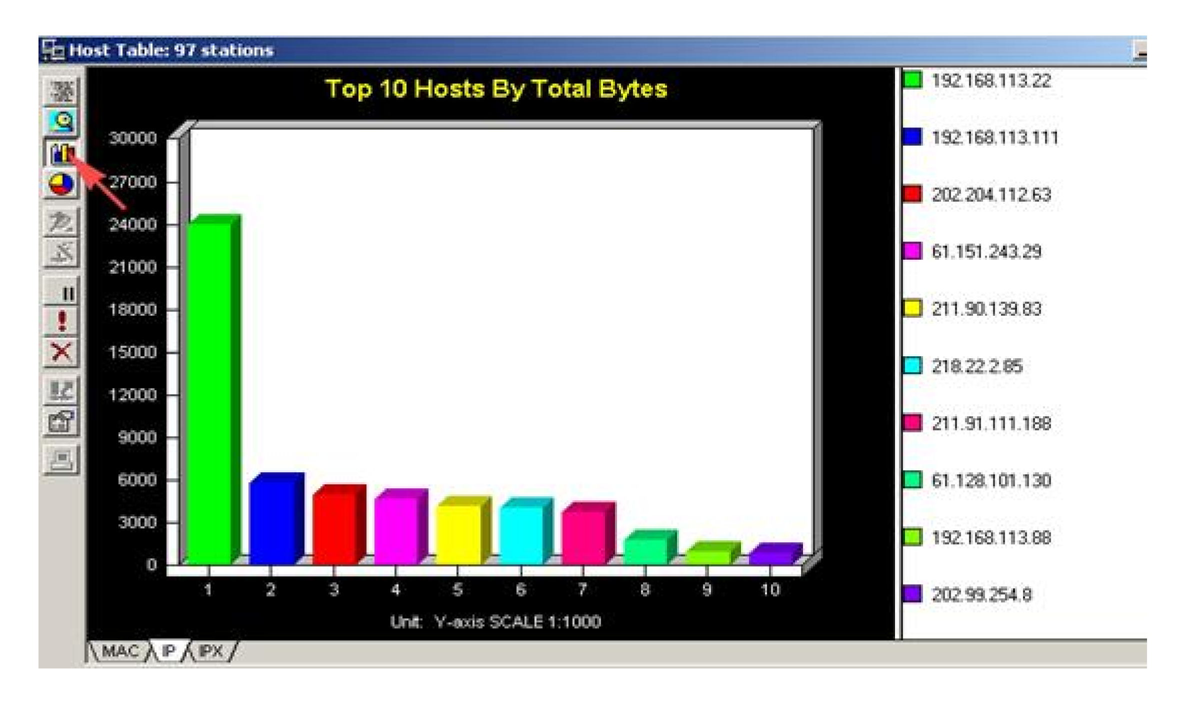


图 6

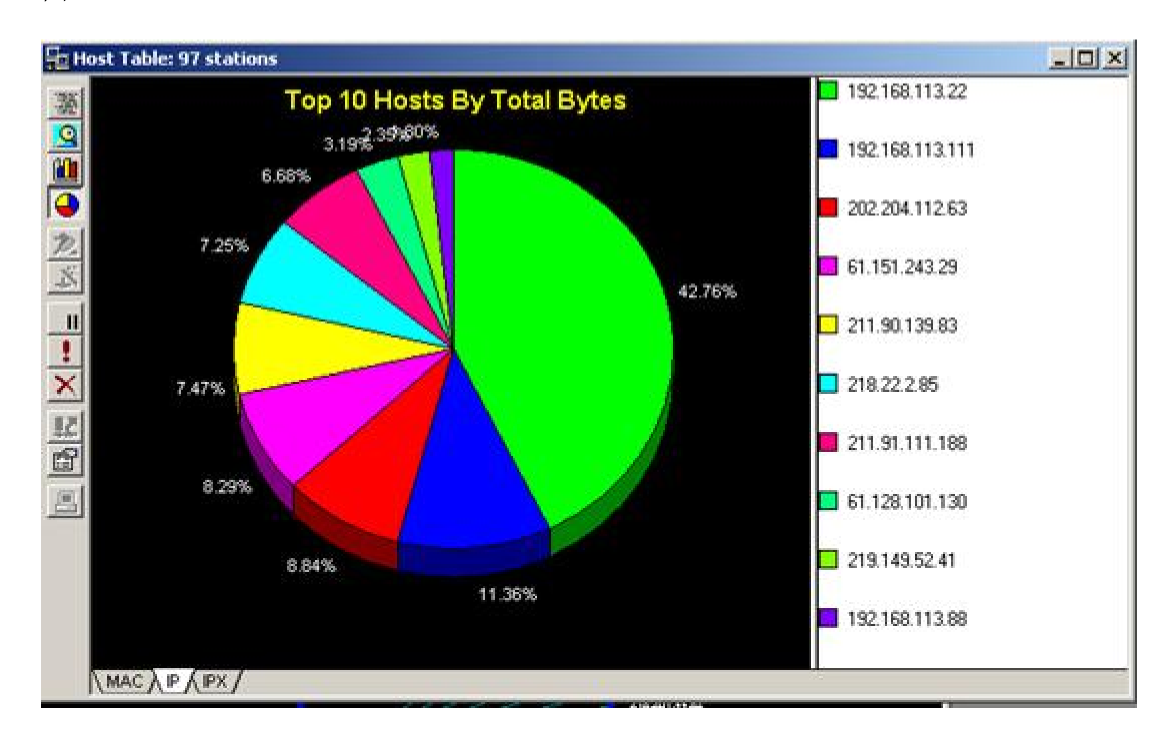


图 7

5、Matrix （网络连接）

点击图 8 中箭头所指的图标，出现全网的连接示意图，图中绿线表示正在发生的网络连接，蓝线表示过去发生的连接。将鼠标放到线上可以看出连接情况。鼠标右键在弹出的菜单中可选择放大（zoom）此图。

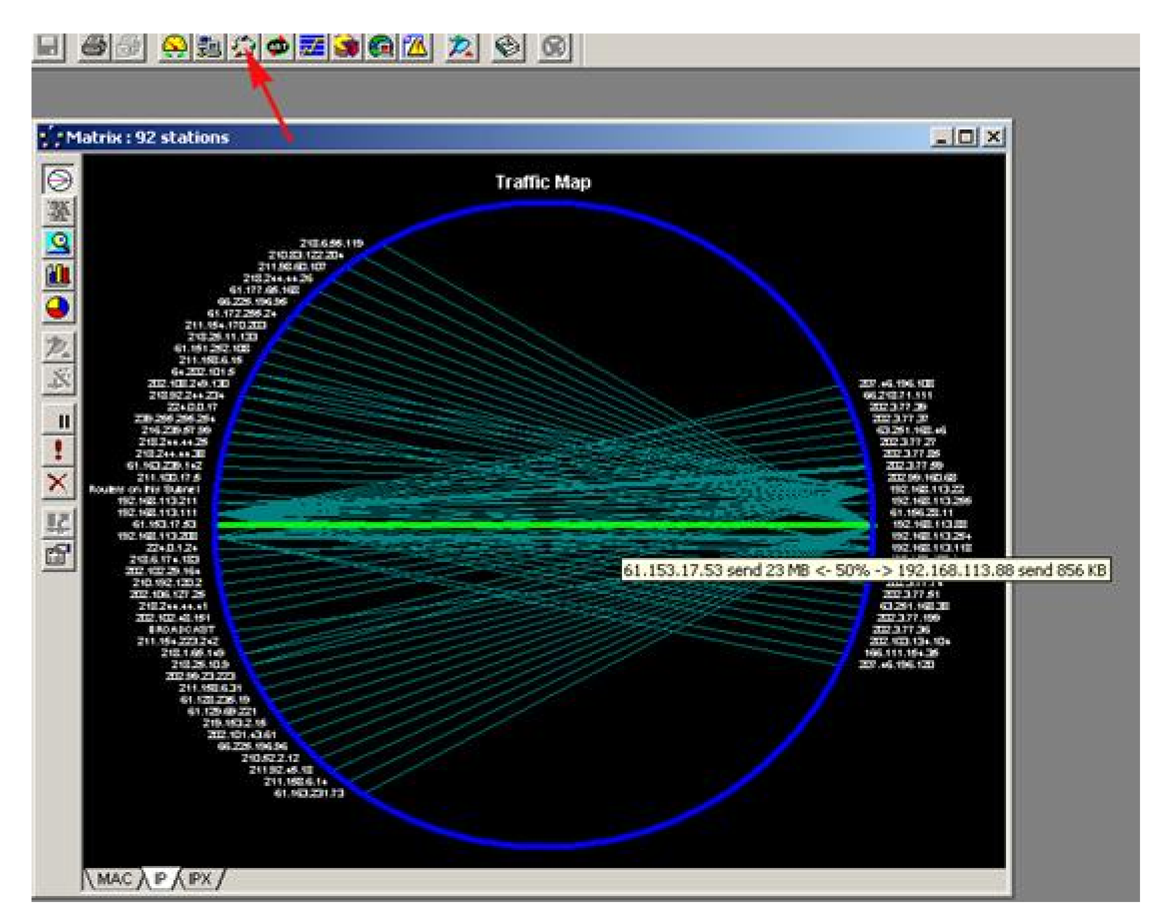


图 8

抓包实例

1、抓某台机器的所有数据包

如图 9 所示，本例要抓 192.168.113.208 这台机器的所有数据包，如图中①选择这台机器。点击②所指图标，出现图 10 界面，等到图 10 中箭头所指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 11 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。

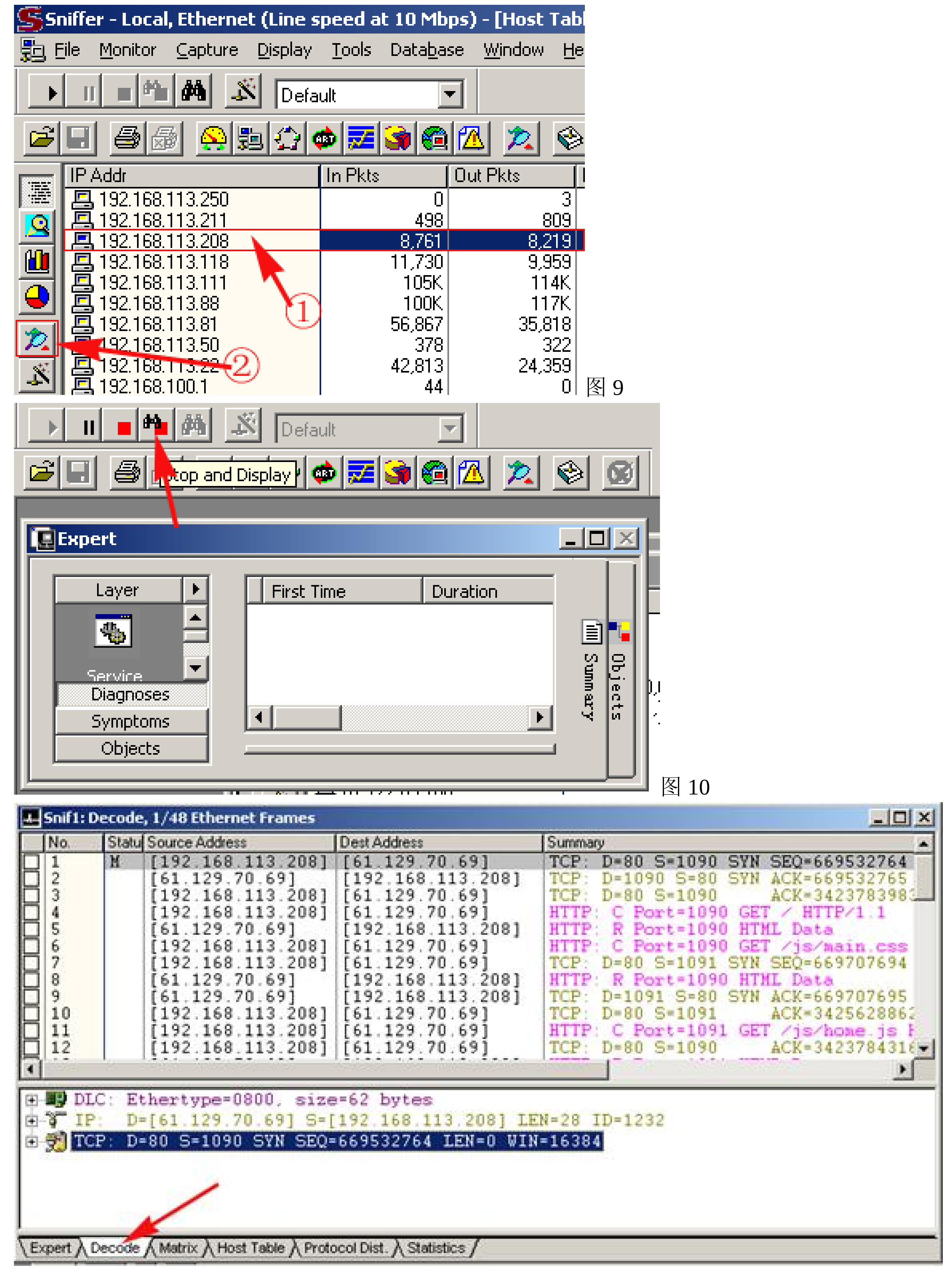


图 9

图 10

图 11

2、抓 Telnet 密码

本例从 192.168.113.208 这台机器 telnet 到 192.168.113.50，用 Sniff Pro 抓到用户名和密码。

步骤 1：设置规则

如图 12 所示，选择 Capture 菜单中的 Defind　Filter，出现图 13 界面，选择图 13 中的 ADDress 项，在 station1 和 station2 中分别填写两台机器的 IP 地址，如图 14 所示选择 Advanced 选项，选择选 IP/TCP/Telnet ,将 Packet Size 设置为 Equal 55， Packet Type 设置为 Normal.。

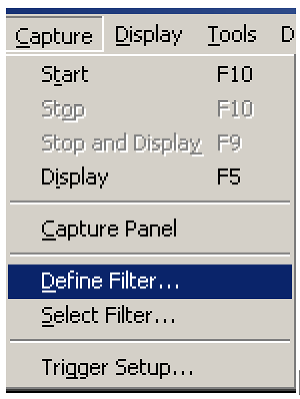


图 12

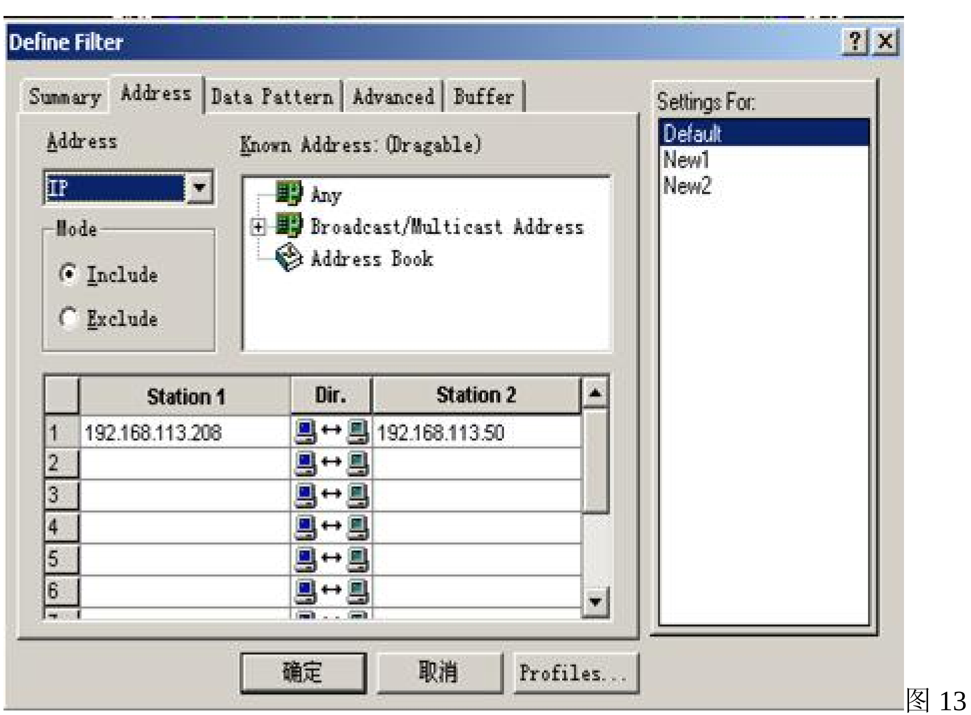


图 13

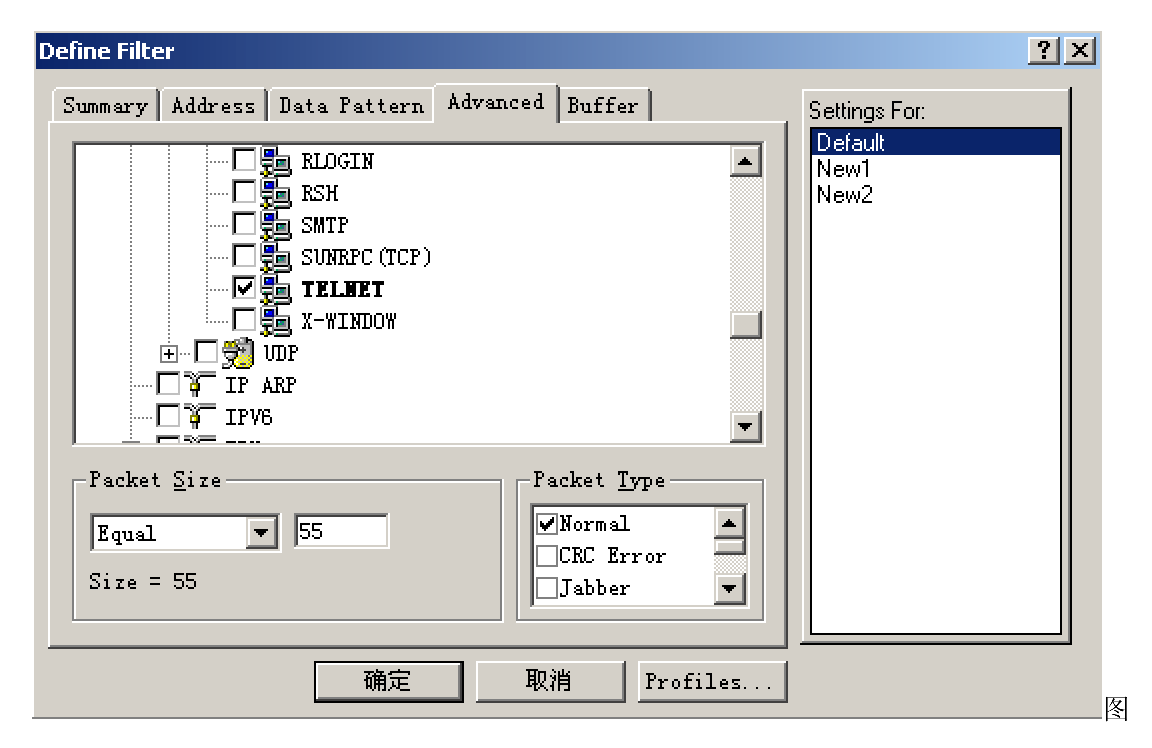


图 14

步骤 2：抓包

按 F10 键出现图 15 界面，开始抓包。

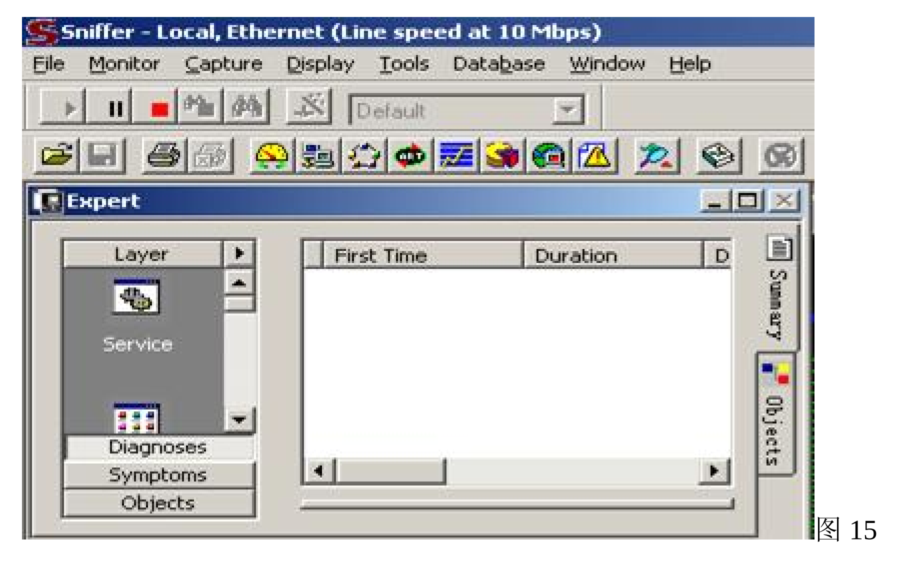


图 15

步骤 3：运行 telnet 命令

本例使 telnet 到一台开有 telnet 服务的 Linux 机器上。

telnet 192.168.113.50

login: test

Password:

步骤 4：察看结果

图 16 中箭头所指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 17 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。可以清楚地看出用户名为 test 密码为 123456。



图 16

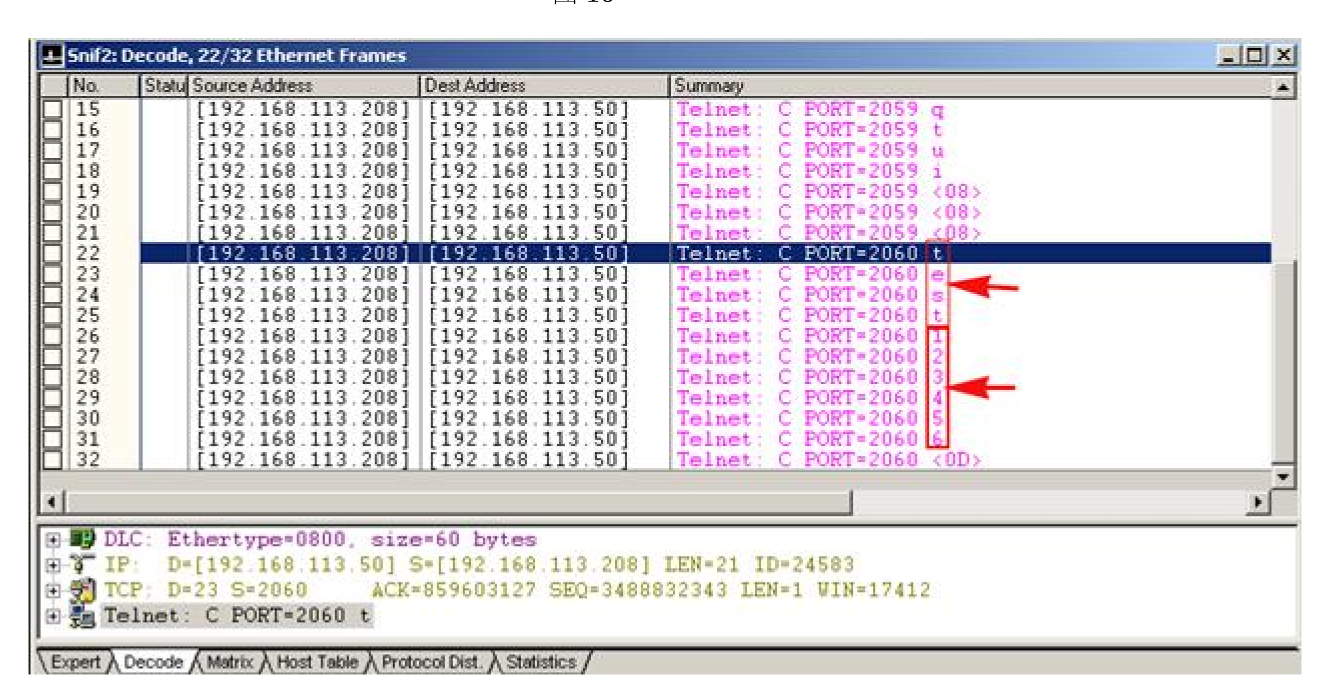


图 17

解释：

虽然把密码抓到了，但大家也许对包大小（Packet Size）设为 55 不理解，网上的数据传送是把数据分成若干个包来传送，根据协议的不同包的大小也不相同，从图 18 可以看出当客户端 telnet 到服务端时一次只传送一个字节的数据，由于协议的头长度是一定的，所以 telnet 的数据包大小=DLC(14 字节)+IP(20 字节)+TCP(20 字节)+数据（一个字节）=55 字节，这样将 Packet Size 设为 55 正好能抓到用户名和密码，否则将抓到许多不相关的包。

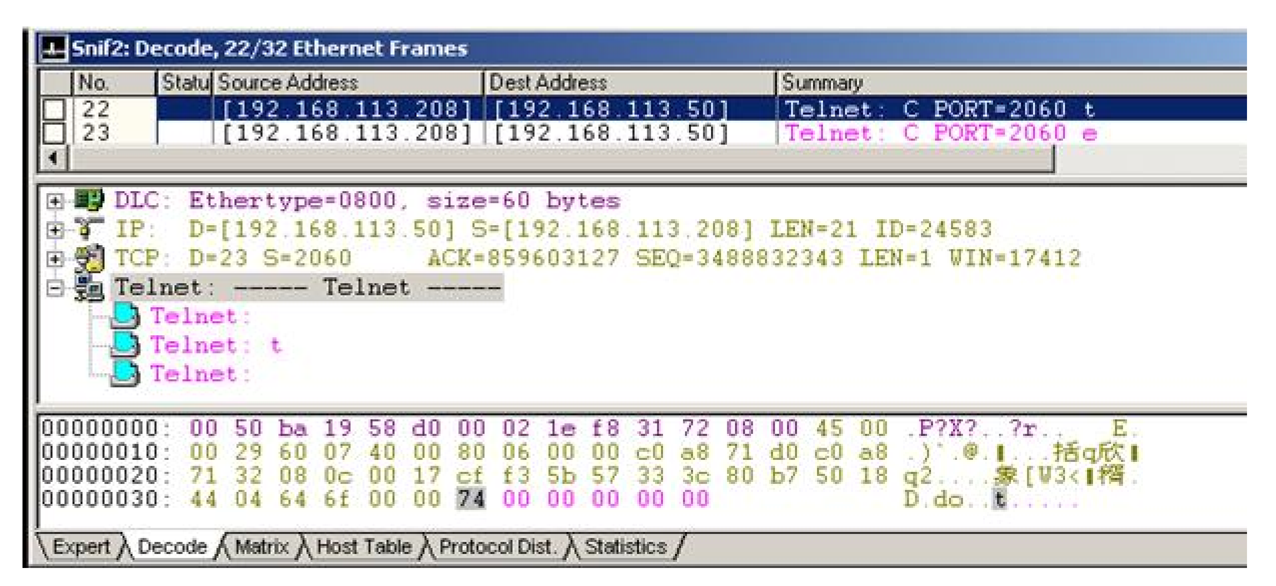


图 18

3、抓 FTP 密码

本例从 192.168.113.208 这台机器 ftp 到 192.168.113.50，用 Sniff Pro 抓到用户名和密码。

步骤 1：设置规则

如图 12 所示，选择 Capture 菜单中的 Defind Filter 出现图 19 界面，选择图 19 中的 ADDress 项，在 station1 和 2 中分别填写两台机器的 IP 地址，选择 Advanced 选项，选择选 IP/TCP/FTP，将 Packet Size 设置为 In Between 63 -71， Packet Type 设置为 Normal。如图 20 所示，选择 Data Pattern 项，点击箭头所指的 Add Pattern 按钮，出现图 21 界面，按图设置 OFFset 为 2F,方格内填入 18，name 可任意起。确定后如图 22 点击 Add NOT 按钮,再点击 Add Pattern 按钮增加第二条规则，按图 23 所示设置好规则，确定后如图 24 所示。

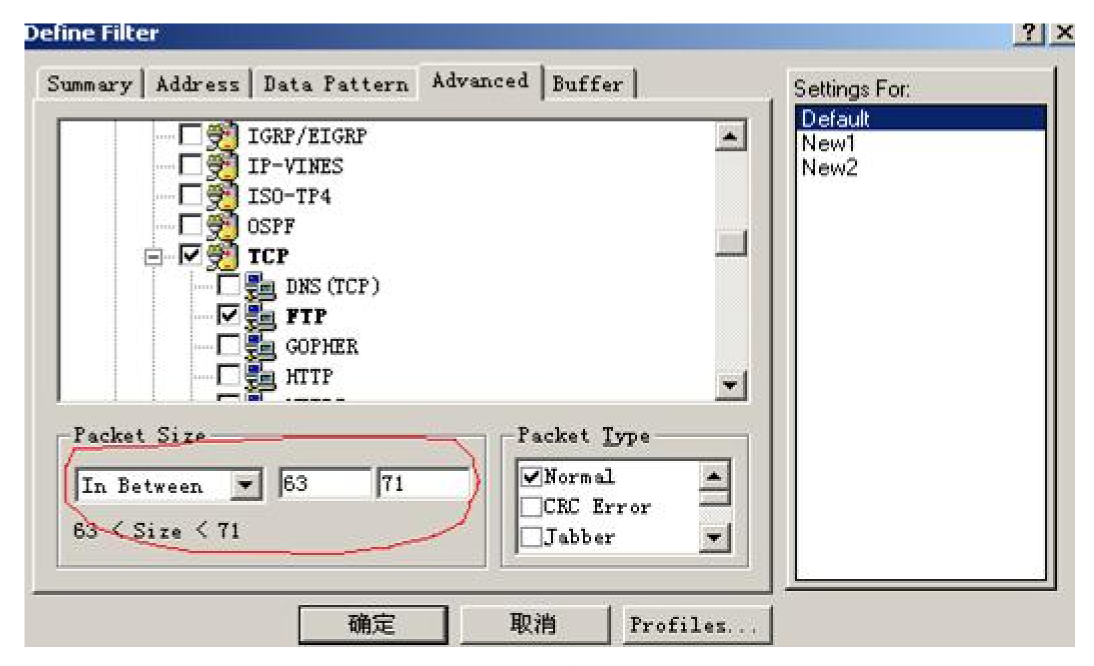

Define Filter
Summary
Address
Data Pattern
Advanced
Buffer
IGRP/EIGRP
IP-VINES
ISO-TP4
OSPF
TCP
DNS (TCP)
FTP
GOPHER
HTTP
Packet Size
In Between
63
71
63 < Size < 71
Packet Type
Normal
CRC Error
Jabber
Settings For:
Default
New1
New2
确定
取消
Profiles...

图 19

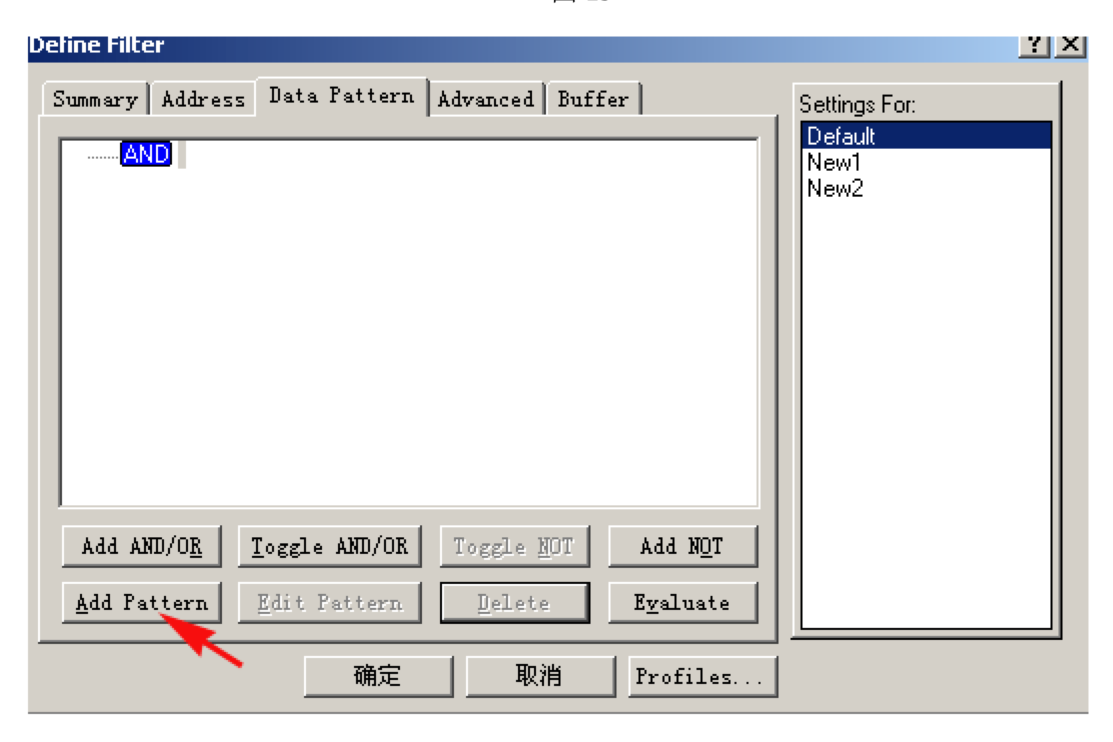

Define Filter
Summary
Address
Data Pattern
Advanced
Buffer
AND
Add AND/OR
Toggle AND/OR
Toggle NOT
Add NOT
Add Pattern
Edit Pattern
Delete
Evaluate
Settings For:
Default
New1
New2
确定
取消
Profiles...

图 20

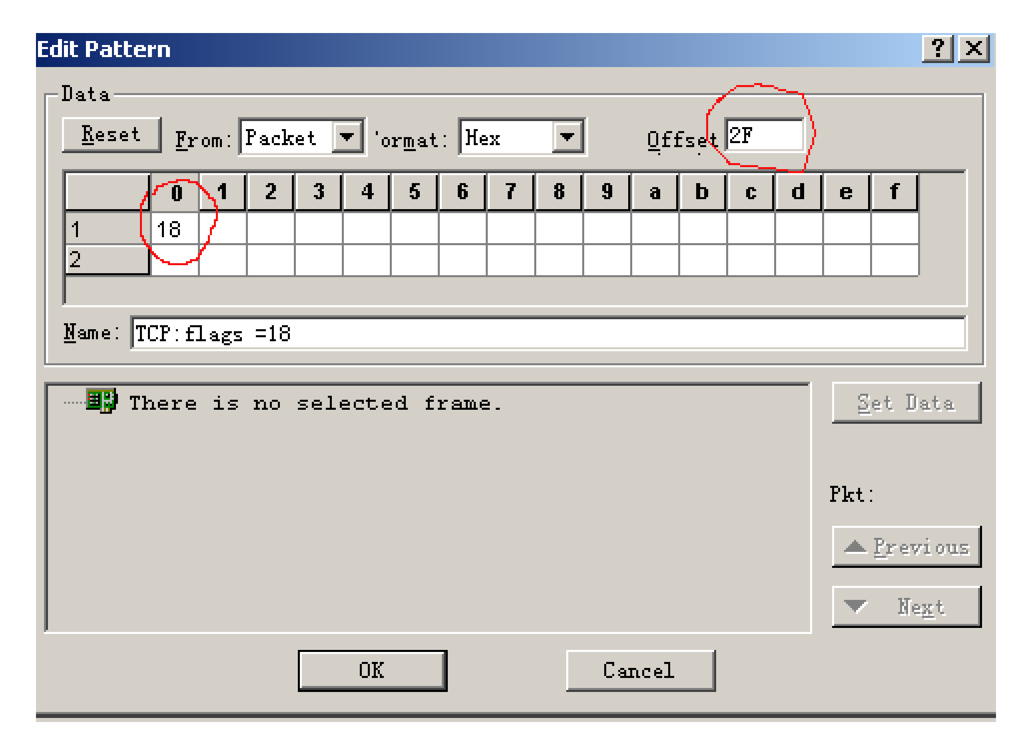

Edit Pattern
Data
Reset
From: Packet
'ormat: Hex
Offset 2F
0 1 2 3 4 5 6 7 8 9 a b c d e f
1 18
2
Name: TCP:flags =18
There is no selected frame.
Set Data
Pkt:
Previous
Next
OK
Cancel

图 21

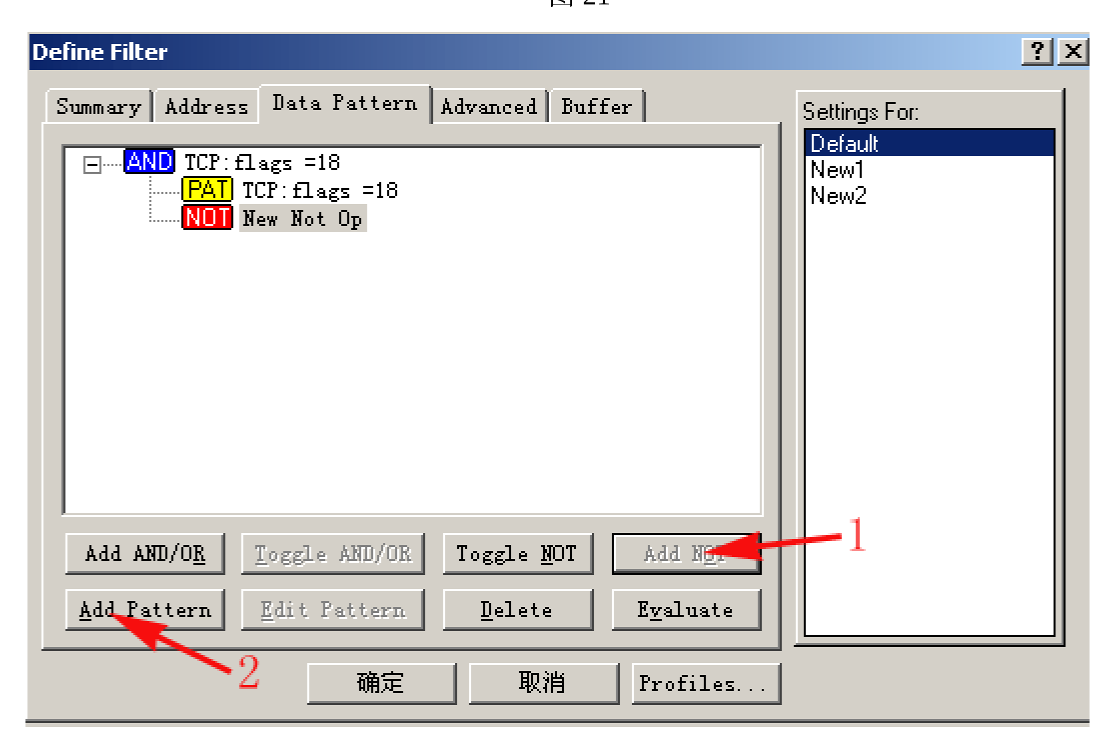

Define Filter
Summary
Address
Data Pattern
Advanced
Buffer
AND TCP:flags =18
PAT TCP:flags =18
NOT New Not Op
Settings For:
Default
New1
New2
Add AND/OR
Toggle AND/OR
Toggle NOT
Add Pattern
Edit Pattern
Delete
Evaluate
1
2
确定
取消
Profiles...

图 22

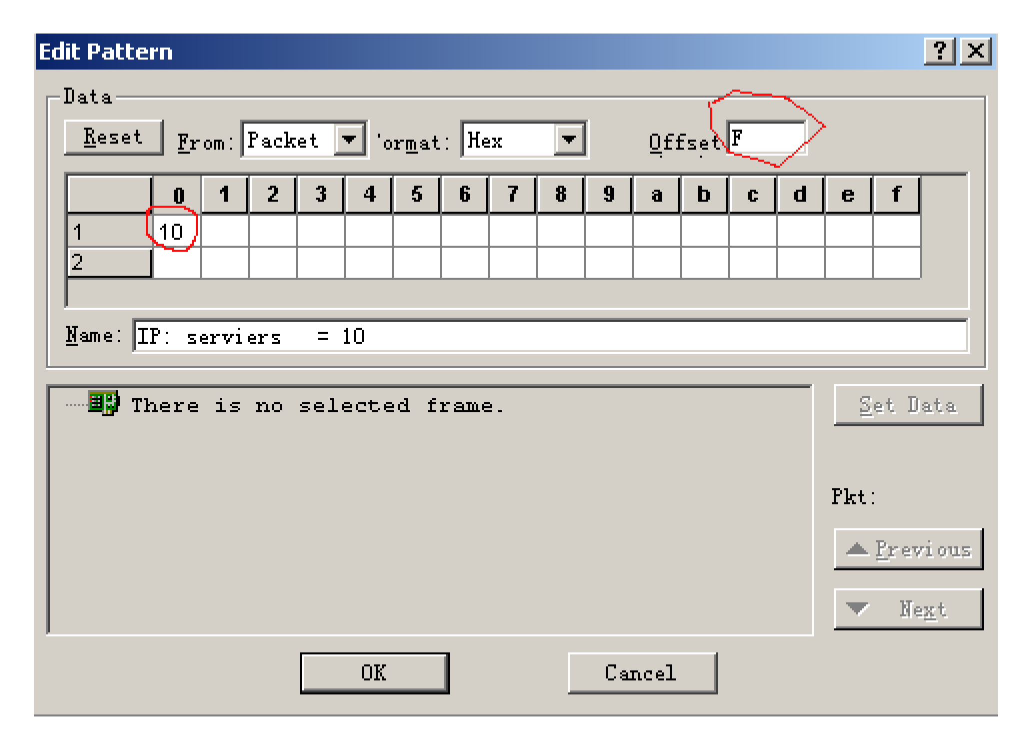


图 23

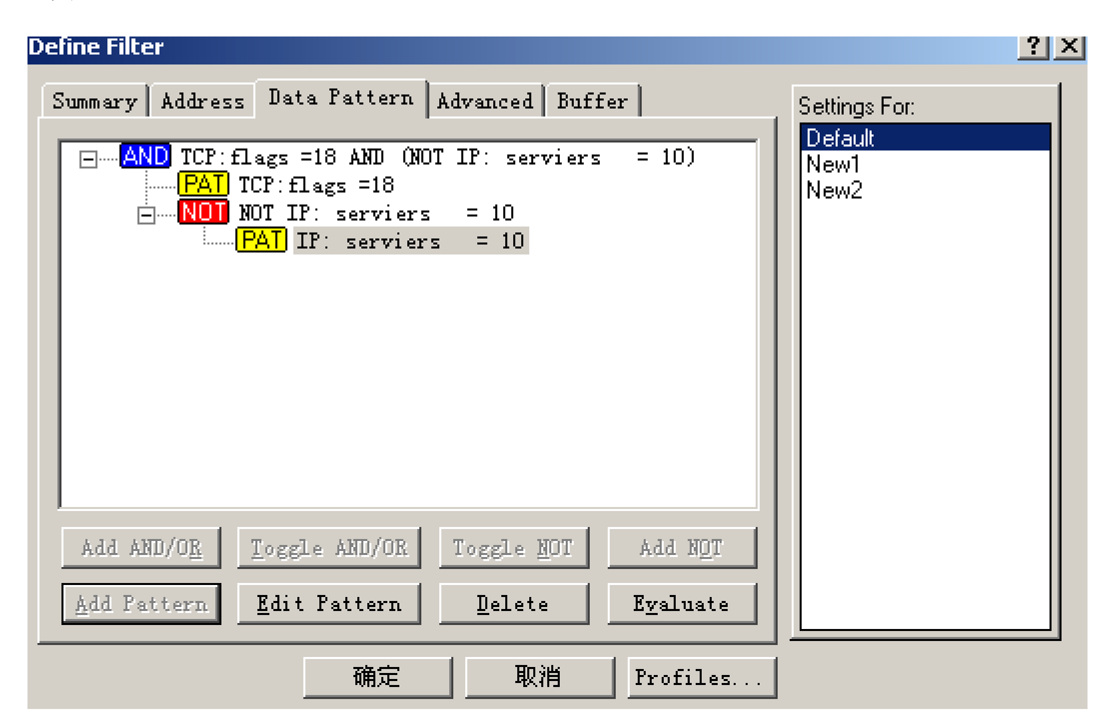


图 24

步骤 2：抓包

按 F10 键出现图 15 界面，开始抓包。

步骤 3：运行 FTP 命令

本例使 FTP 到一台开有 FTP 服务的 Linux 机器上

D:/>ftp 192.168.113.50

Connected to 192.168.113.50.

220 test1 FTP server (Version wu-2.6.1(1) Wed Aug 9 05:54:50 EDT 2000) ready.

User (192.168.113.50:(none)): test

331 Password required for test.

Password:

步骤 4：察看结果

图 16 中箭头所指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 25 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。可以清楚地看出用户名为 test 密码为 123456789。

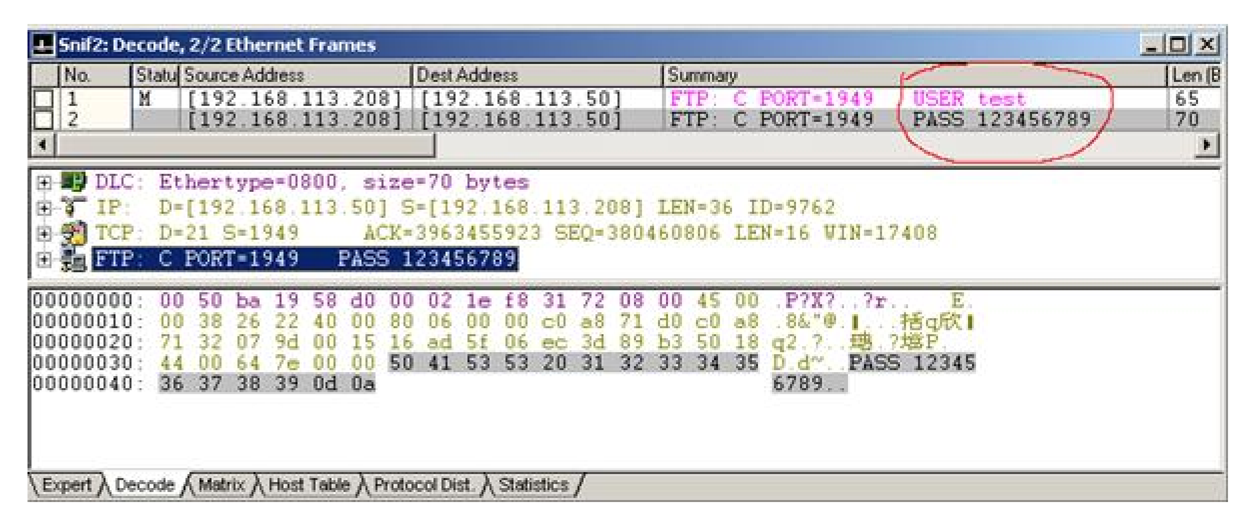


图 25

解释：

虽然把密码抓到了，但大家也许设不理解，将图 19 中 Packet Size 设置为 63 -71 是根据用户名和口令的包大小来设置的，图 25 可以看出口令的数据包长度为 70 字节，其中协议头长度为：14+20+20=54，与 telnet 的头长度相同。Ftp 的数据长度为 16，其中关键字 PASS 占 4 个字节，空格占 1 个字节，密码占 9 个字节，Od 0a(回车 换行)占 2 个字节，包长度=54+16=70。如果用户名和密码比较长那么 Packet Size 的值也要相应的增长。

Data Pattern 中的设置是根据用户名和密码中包的特有规则设定的，为了更好的说明这个问题，请在开着图 15 的情况下选择 Capture 菜单中的 Defind Filter，如图 20 所示，选择 Data Pattern 项，点击箭头所指的 Add Pattern 按钮，出现图 26 界面，选择图中 1 所指然后点击 2 所指的 Set Data 按钮。OFFset、方格内、Name 将填上相应的值。

同理图 27 中也是如此。

这些规则的设置都是根据你要抓的包的相应特征来设置的，这些都需要对 TCP/IP 协议的深入了解，从图 28 中可以看出网上传输的都是一位一位的比特流，操作系统将比特流转换为二进制，Sniffer 这类的软件又把二进制换算为 16 进制，然后又为这些数赋予相应的意思，图中的 18 指的是 TCP 协议中的标志位是 18。OFFset 指的是数据包中某位数据的位置，方格内填的是值。

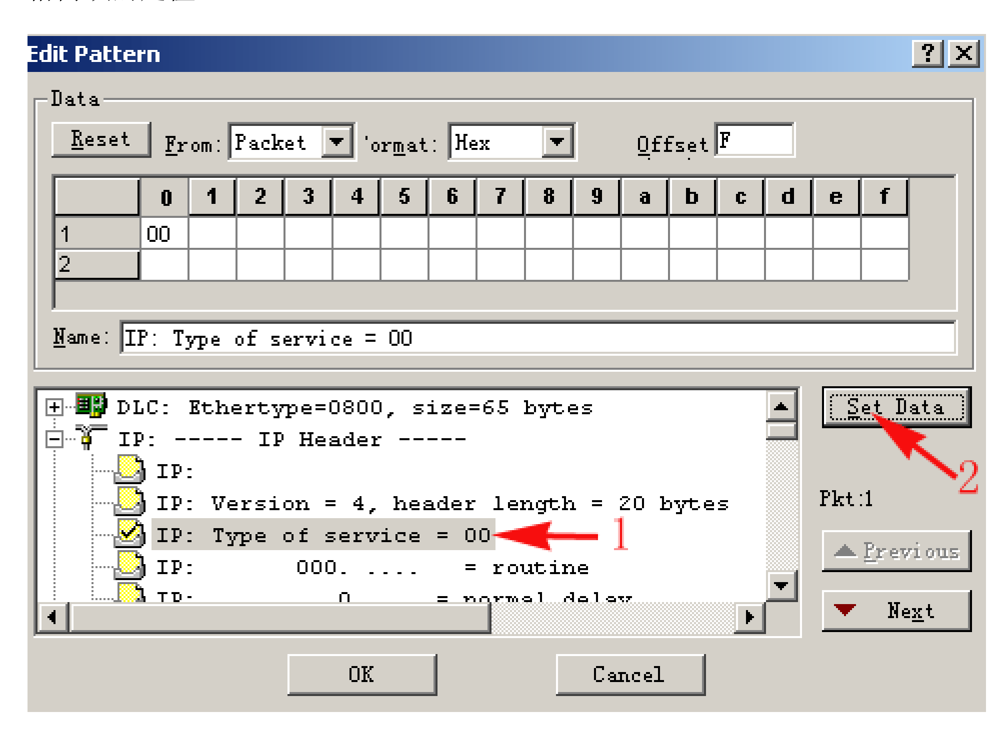


图 26

图 27

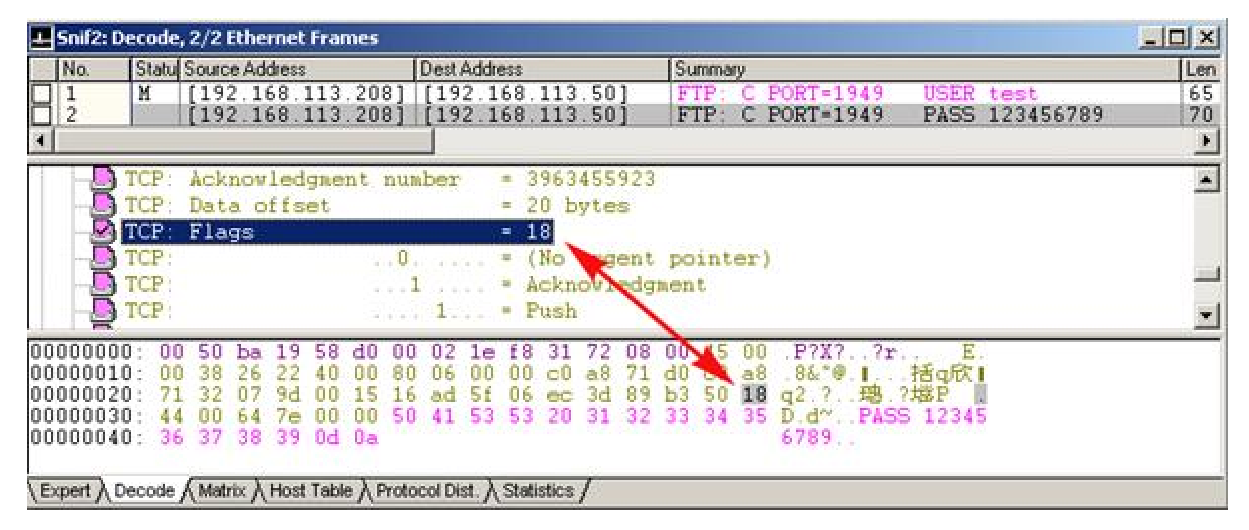


图 28

4、抓 HTTP 密码

步骤 1：设置规则

按照下图 29、30 进行设置规则，设置方法同上。

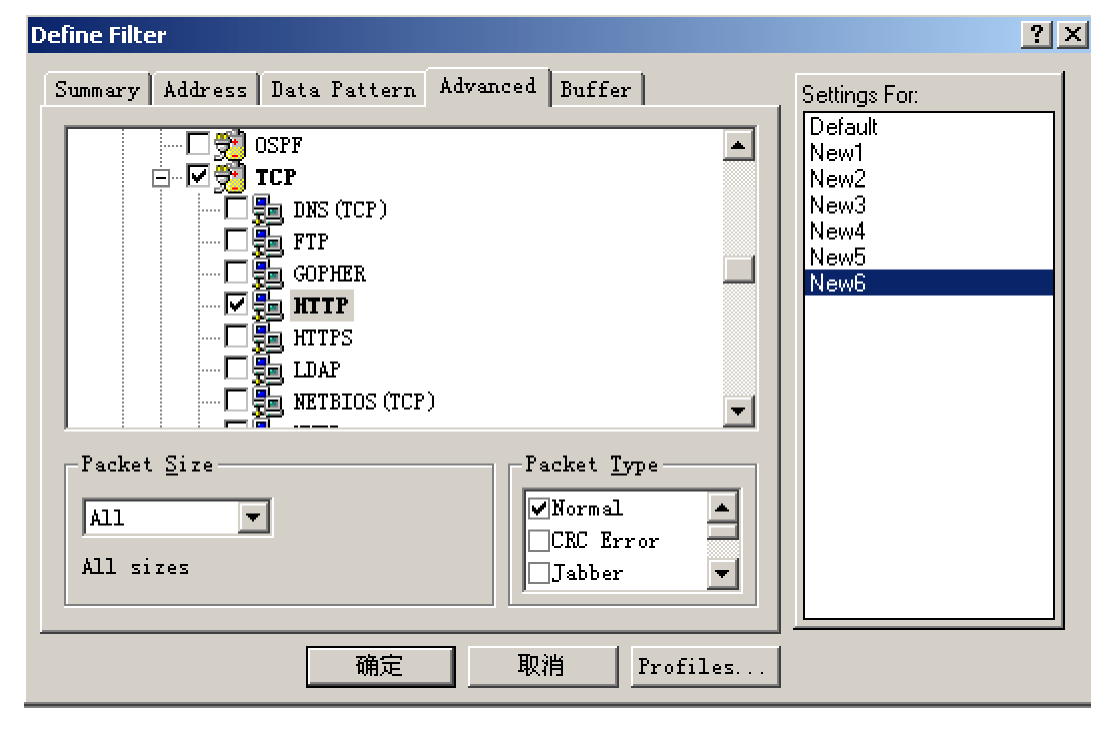


图 29

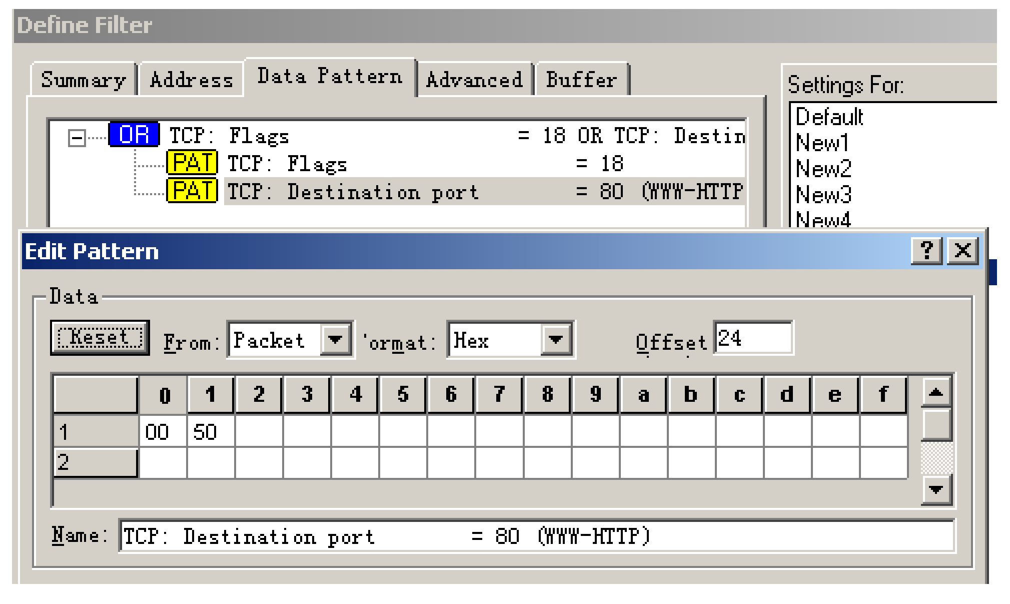


图 30

步骤 2：抓包

按 F10 键开始抓包。

步骤 3：访问www.ccidnet.com网站

步骤 4：察看结果

图 16 中箭头所 指的望远镜图标变红时，表示已捕捉到数据，点击该图标出现图 31 界面，选择箭头所指的 Decode 选项即可看到捕捉到的所有包。在 Summary 中找到含有 POST 关键字的包，可以清楚地看出用户名为 qiangkn997，密码为?，这可是我邮箱的真实密码！当然不能告诉你，不过欢迎来信进行交流。

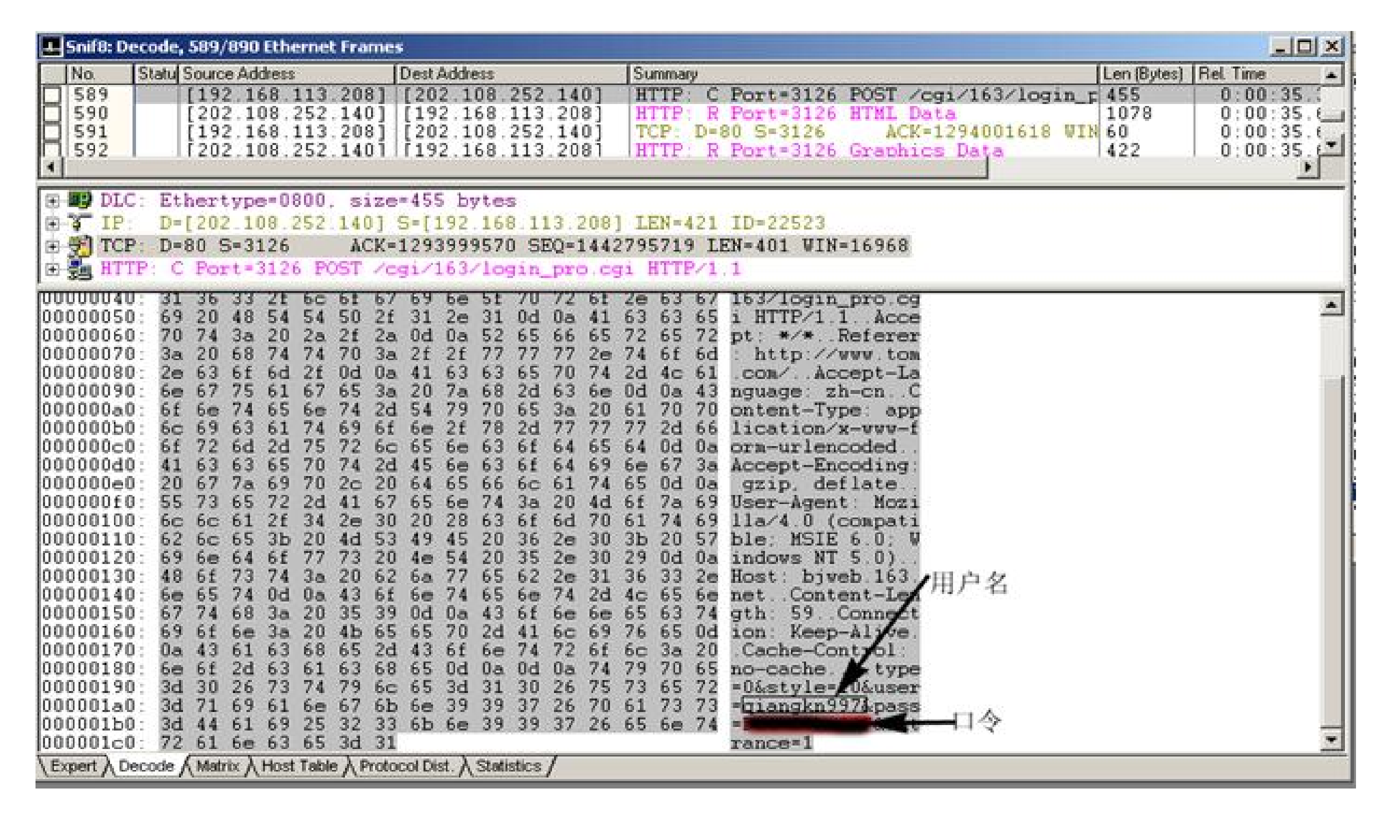


图 31

后记

本文中的例子是网内试验，若捕捉全网机器的有关数据请将图 13 中的 station 设置为 any<->any，作为学习研究可以，可别做坏事！如果要用好 Sniff Pro 必须有扎实的网络基础知识特别是 TCP/IP 协议的知识，其实 Sniff Pro 本身也是学习这些知识的好工具。

Sniffer Pro 是个博大精深的工具，由于水平有限，本文这是介绍了其中的一小部分，希望

Sniffer 使用简介（二）

一、捕获数据包前的准备工作

在默认情况下，sniffer 将捕获其接入碰撞域中流经的所有数据包，但在某些场景下，有些数据包可能不是我们所需要的，为了快速定位网络问题所在，有必要对所要捕获的数据包作过滤。Sniffer 提供了捕获数据包前的过滤规则的定义，过滤规则包括 2、3 层地址的定义和几百种协议的定义。定义过滤规则的做法一般如下：

1、在主界面选择 capture→definefilter 选项。

2、definefilter→address，这是最常用的定义。其中包括 MAC 地址、ip 地址和 ipx 地址的定义。以定义 IP 地址过滤为例，见图 1。

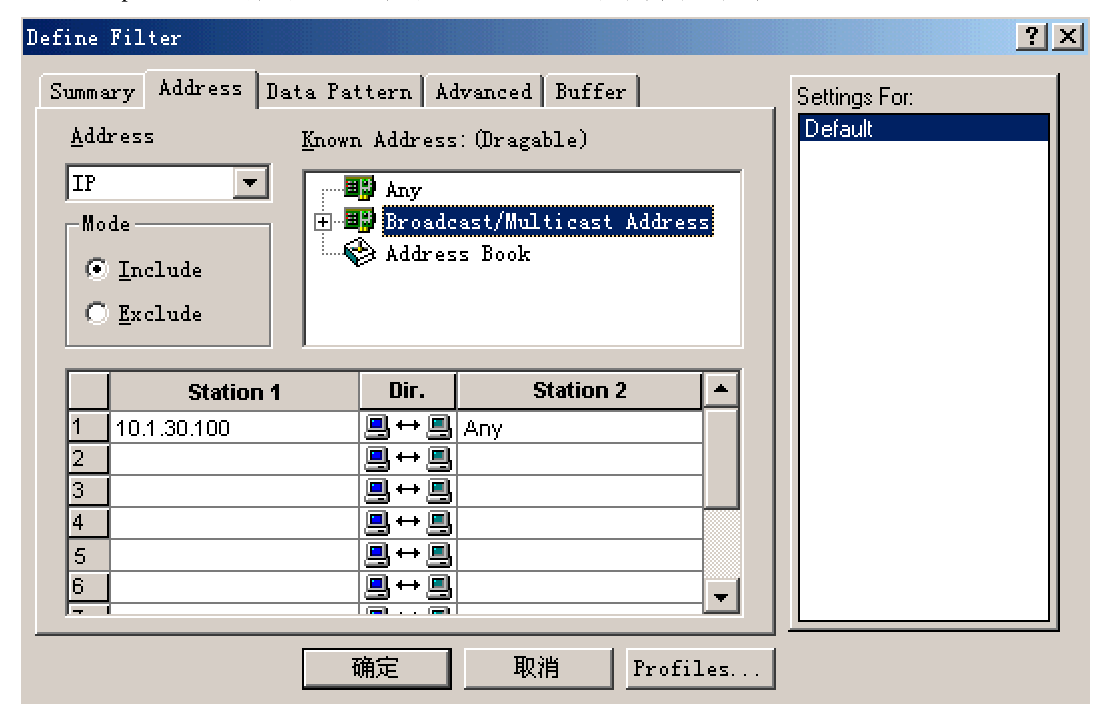
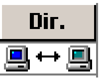


图 1

比如，现在要捕获地址为 10.1.30.100 的主机与其他主机通信的信息，在 Mode 选项卡中，选 Include(选 Exclude 选项，是表示捕获除此地址外所有的数据包)；在 station 选项中，在任意一栏填上 10.1.30.100，另外一栏填上 any(any 表示所有的 IP 地址）。这样就完成了地址的定义。

注意到 Dir. 栏的图标：



表示，捕获 station1 收发的数据包；



表示，捕获 station1 发送的数据包；



表示，捕获 station1 收到的数据包。

最后，选取

Profiles...

将定义的规则保存下来，供以后使用。

3、definefilter→advanced，定义希望捕获的相关协议的数据包。如图 2。

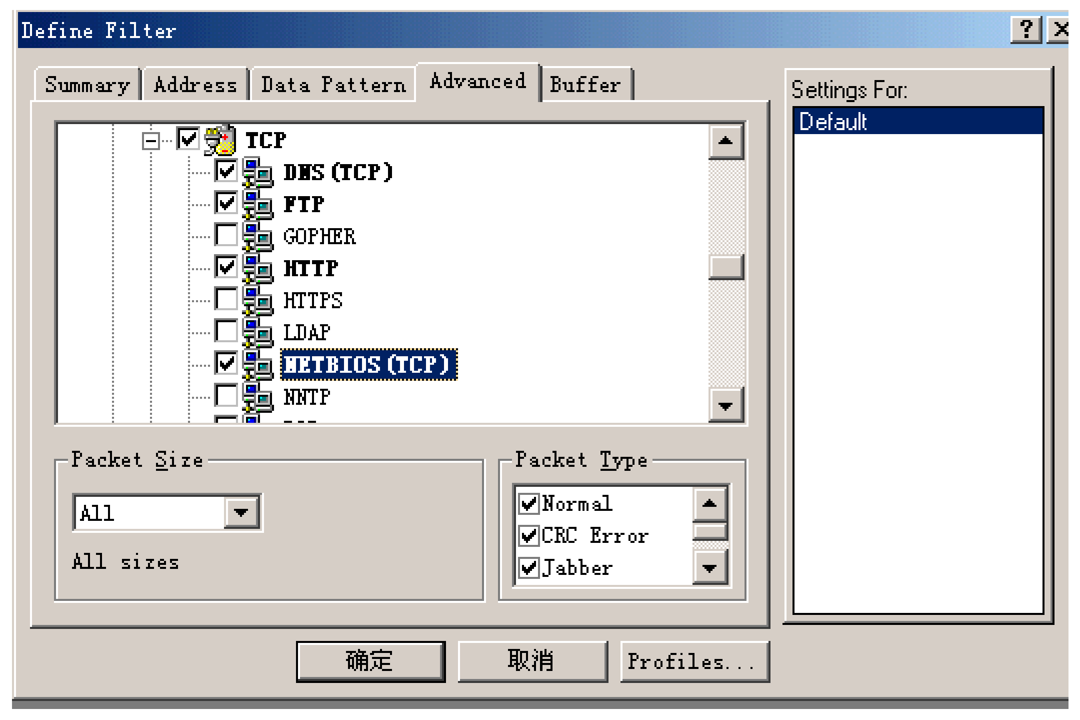


图 2

比如，想捕获 FTP、NETBIOS、DNS、HTTP 的数据包，那么说首先打开 TCP 选项卡，再进一步选协议；还要明确 DNS、NETBIOS 的数据包有些是属于 UDP 协议，故需在 UDP 选项卡做类似 TCP 选项卡的工作，否则捕获的数据包将不全。如果不选任何协议，则捕获所有协议的数据包。

PacketSize 选项中，可以定义捕获的包大小，图 3，是定义捕获包大小界于 64 至 128bytes 的数据包。

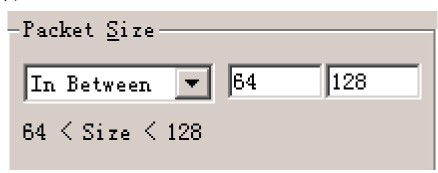


图 3

4、definefilter→buffer,定义捕获数据包的缓冲区。如图 4:

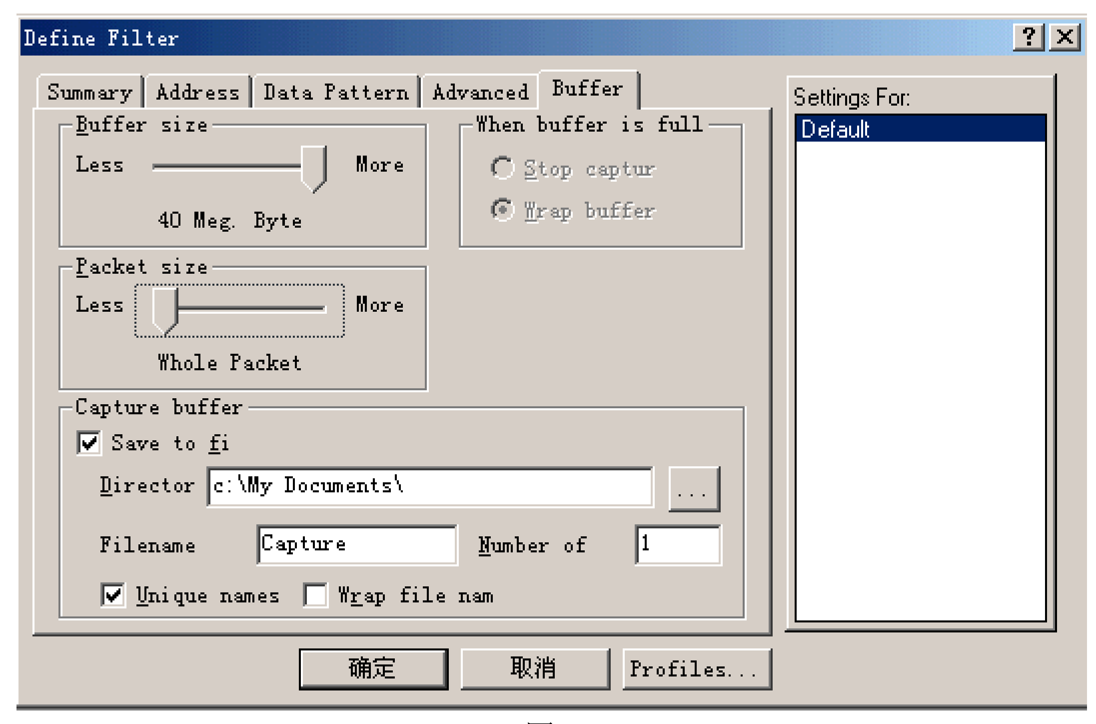


图 4

Buffersize 选项卡，将其设为最大 40M。

Capturebuffer 选项卡，将设置缓冲区文件存放的位置。

5、最后，需将定义的过滤规则应用于捕获中。如图 5:

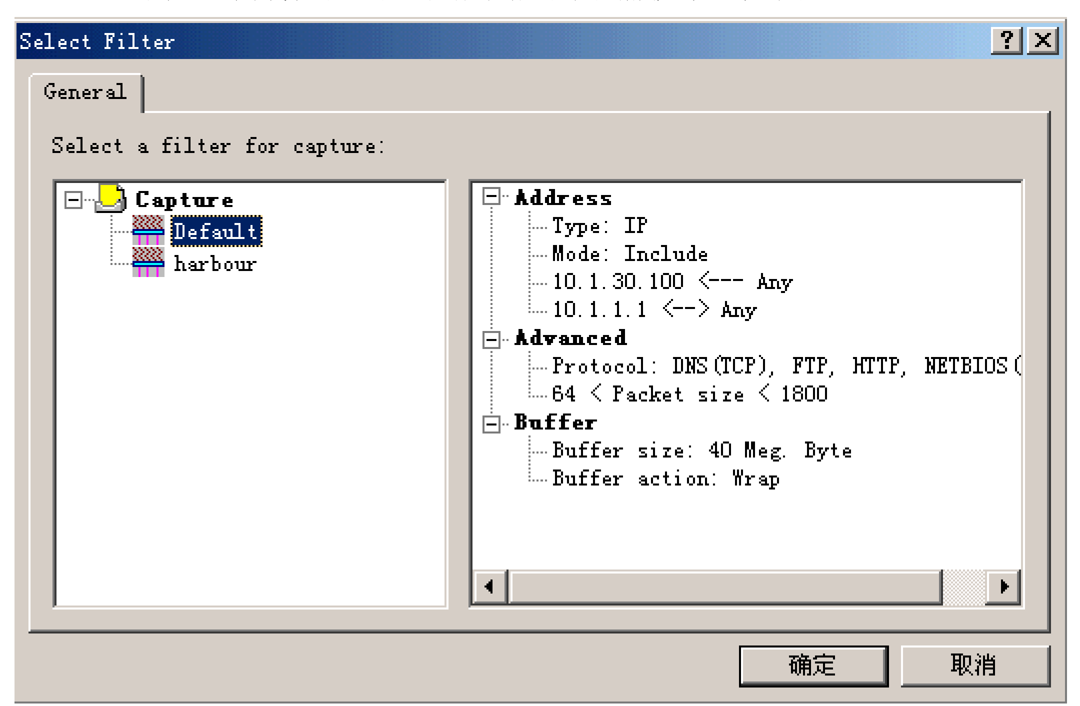


图 5

点选 SelectFilter→Capture 中选取定义的捕获规则

Sniffer 使用简介（下）

二、　捕获数据包时观察到的信息

Capture→Start,启动捕获引擎。

sniffer 可以实时监控主机、协议、应用程序、不同包类型等的分布情况。如图 6：

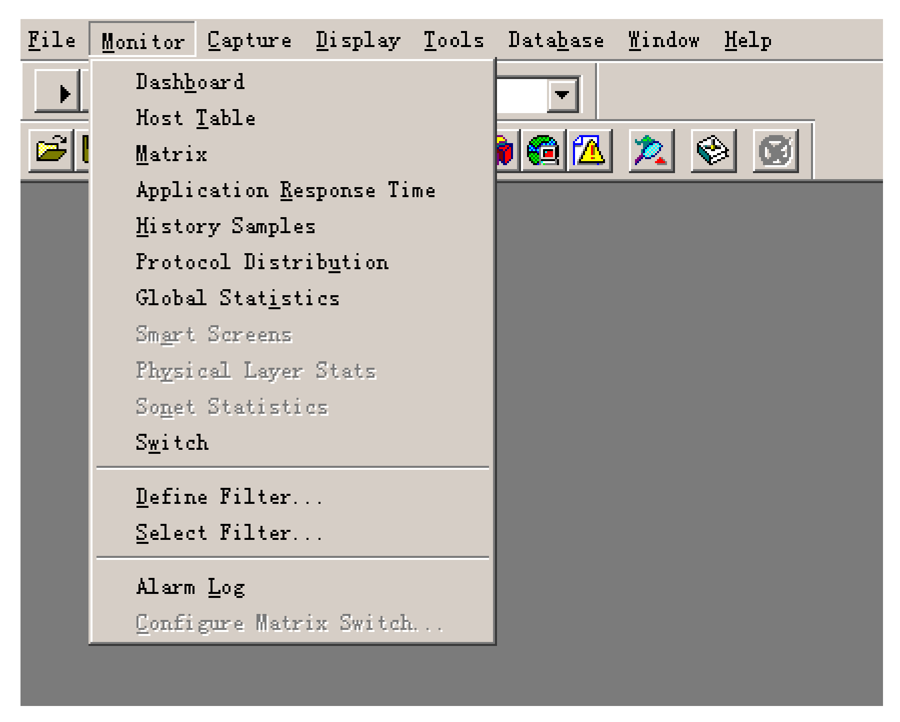


图 6

Dashboard:可以实时统计每秒钟接收到的包的数量、出错包的数量、丢弃包的数量、广播包的数量、多播包的数量以及带宽的利用率等。

HostTable：可以查看通信量最大的前 10 位主机。

Matrix:通过连线，可以形象的看到不同主机之间的通信。

ApplicationResponseTime:可以了解到不同主机通信的最小、最大、平均响应时间方面的信息。

HistorySamples:可以看到历史数据抽样出来的统计值。

Protocoldistribution:可以实时观察到数据流中不同协议的分布情况。

Switch:可以获取 cisco 交换机的状态信息。

在捕获过程中，同样可以对想观察的信息定义过滤规则，操作方式类似捕获前的过滤规则。

三、捕获数据包后的分析工作

要停止 sniffer 捕获包时，点选 Capture→Stop 或者 Capture→StopandDisplay,前者停止捕获包,后者停止捕获包并把捕获的数据包进行解码和显示。如图 7：

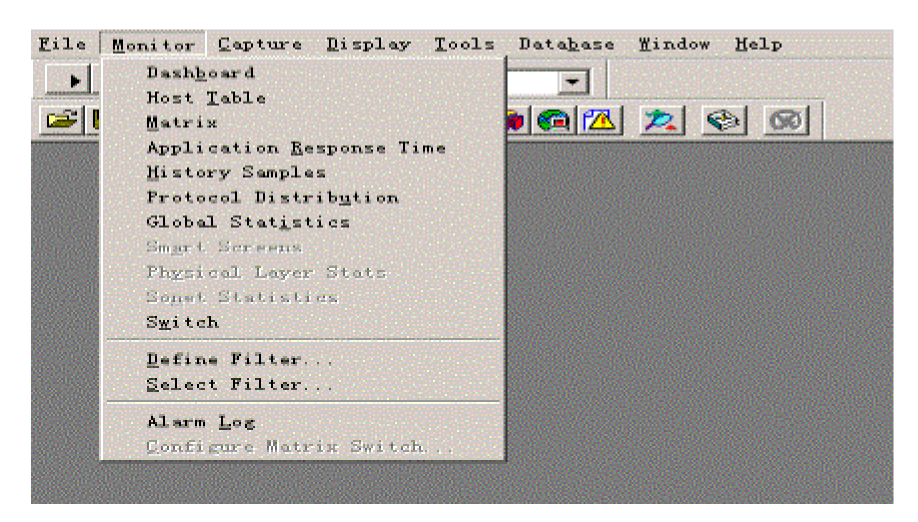


图 7

Decode:对每个数据包进行解码，可以看到整个包的结构及从链路层到应用层的信息，事实上，sniffer 的使用中大部分的时间都花费在这上面的分析，同时也对使用者在网络的理论及实践经验上提出较高的要求。素质较高的使用者借此工具便可看穿网络问题的结症所在。

Expert:这是 sniffer 提供的专家模式，系统自身根据捕获的数据包从链路层到应用层进行分类并作出诊断。其中 diagnoses 提出非常有价值的诊断信息。图 8，是 sniffer 侦查到 IP 地址重叠的例子及相关的解析。

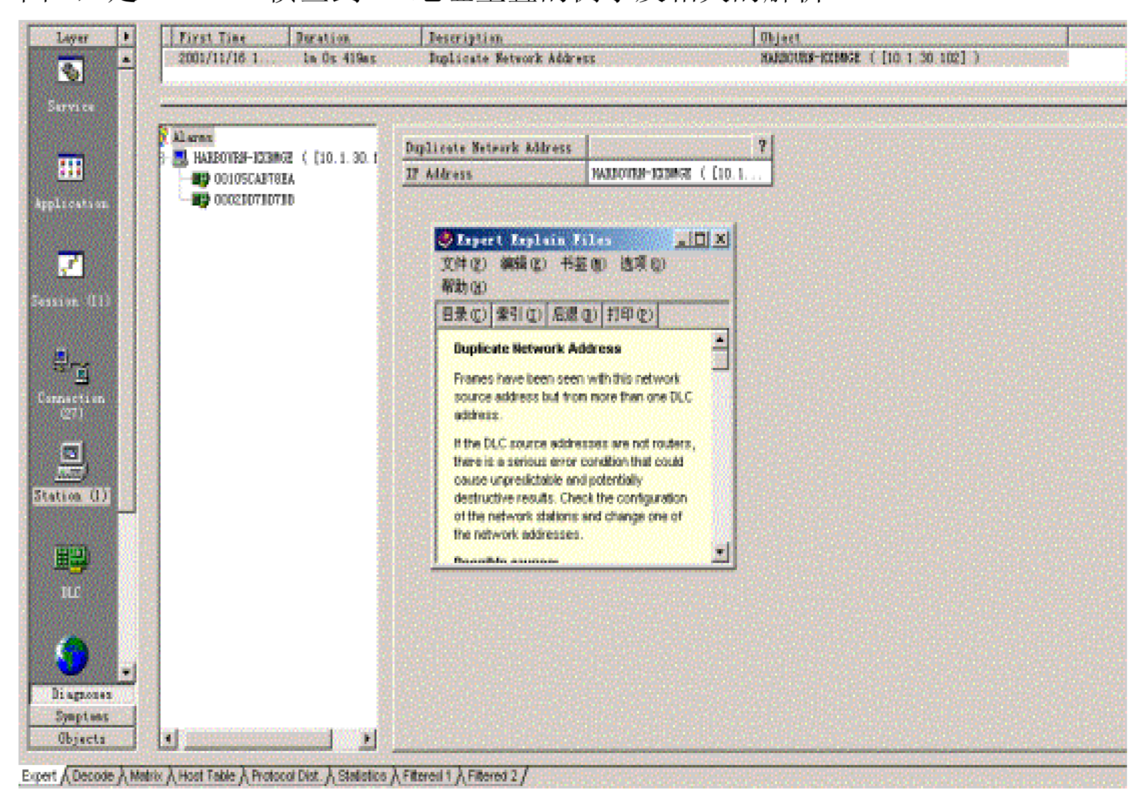


图 8

sniffer 同样提供解码后的数据包过滤显示。
要对包进行显示过滤需切换到 Decode 模式。

Displayàdefinefilter，定义过滤规则。

Displayàselectfilter,应用过滤规则。

显示过滤的使用基本上跟捕获过滤的使用相同。

四、sniffer 提供的工具应用

sniffer 除了提供数据包的捕获、解码及诊断外，还提供了一系列的工具，包括包发生器、ping、traceroute、DNSlookup、finger、whois 等工具。

其中，包发生器比较有特色，将做简单介绍。其他工具在操作系统中也有提供，不做介绍。

包发生器提供三种生成数据包的方式：

点选

，新构一个数据包，包头、包内容及包长由用户直接填写。图 9，定义一个广播包，使其连续发送，包的发送延迟位 1ms

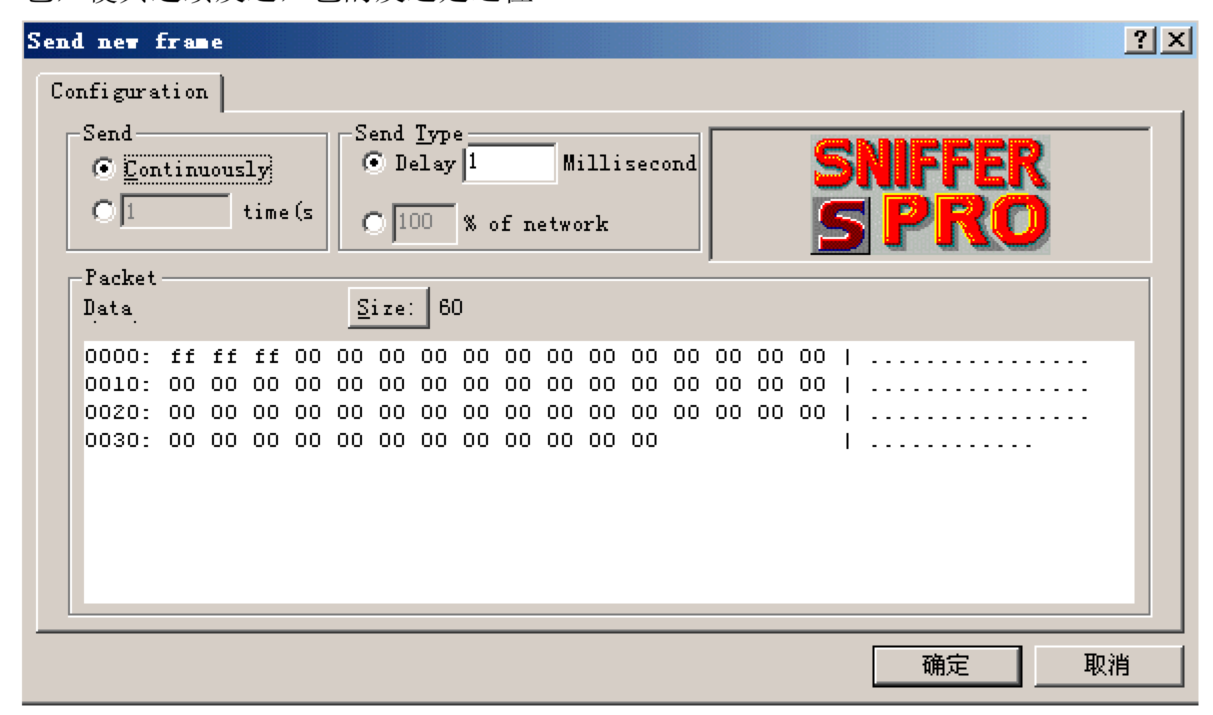


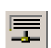
图 9

点选



发送在 Decode 中所定位的数据包，同时可以在此包的基础上对数据包进行如前述的修改。

点选



，发送 buffer 中所有的数据包，实现数据流的重放。见图 10：



图 10

可以定义连续地发送 buffer 中地数据包或只发送一次 buffer 中地数据包。请特别注意，不要在运行的网络中重放数据包，否则容易引起严重的网络问题。数据包的重放经常用于实验环境中。